Chapter 1
Introduction to PDE-constrained optimisation

The use of computational models based on the numerical solution of partial differential equations (PDEs) to simulate physical processes is a powerful complement to physical experiments. Simulations can be undertaken to consider scenarios for which experiments are impossible, such as climate physics or the dynamics of black holes and galaxies. However, a particular strength of simulations is in answering the inverse problems which pervade science and engineering: to which inputs of my system are my output results most sensitive? Is this system stable or unstable? Which configuration or design produces the best outcome? This last question can be rephrased as the core problem of PDE-constrained optimisation: which input parameters minimise some output measurement, given the constraint that the system state must be a solution to a given PDE.

In engineering, the need for optimisation emerges regularly when seeking improved designs. A classical example is concerned with a key element in aeronautical engineering:

What is the optimal shape of an aerofoil?

In the pioneering work of [20], this question is considered as an optimisation problem governed by the Euler equations for compressible flow. A similar design problem is investigated in Chap. 3 of this work. It considers the situation where a large number of tidal turbines are to be deployed within an array in order to extract energy from tidal currents. The investigated optimisation problem is:

What is the optimal spatial distribution of the tidal turbines?

A common task in the geosciences is to determine unknown parameters such that a computer model best reproduces existing measurements [13]. For example, satellite imagery can provide detailed surface information about the ocean, but in general little is known about its interior [35]. The ECCO2 project incorporates most oceanographic and meteorologic measurements available to date into an ocean simulator to create an accurate description of the time-evolving state of the ocean [25]. Here, the problem formulation is:

T. Schwedes et al., *Mesh Dependence in PDE-Constrained Optimisation*, Mathematics of Planet Earth,
DOI 10.1007/978-3-319-59483-5_1

What is the state of the ocean at the beginning of the observation interval that minimises the difference between simulation and measurements?

Despite the variety of these tasks, they can all be formulated as optimisation problems constrained by PDEs. In this work we will employ the finite element method to produce numerical solutions, so the PDEs will be stated in weak, or variational, form. The generic form of a PDE-constrained optimisation problem is then:

$$\begin{aligned} &\min_{\substack{u\in V\\ m\in M}} J(u,m) \\ &\text{subject to } F(u,m;v)=0 \quad \forall v\in V', \end{aligned} \tag{1.1}$$

where M is the space of all possible values of optimisation parameters, and V, V' are suitable function spaces – indeed in most cases $V = V'$. Further, $J : M \otimes V \to \mathbb{R}$ is the functional of interest that is to be minimised, and $F(u,m;v) = 0$ is a PDE parametrised by m with solution u for all values of the test function v.

As an example, reconsider the optimal design of the aerofoil. In this case, the parameter m contains the parametrised shape of the aerofoil, for example the coefficients of its Bézier curve. The physics are described by the Euler equations, which yield the pressure and velocity for a given aerofoil design. Finally, the functional of interest J computes some performance metric for the aerofoil, for example by evaluating the drag-lift ratio. The issue where a performance metric is to be maximised while formulation (1.1) actually describes a minimisation problem is easily resolved by seeking to minimise the negative of J instead.

The necessary ingredients for solving such optimisation problems numerically are the following: first, we need to be able to approximate the solution u of $F(u,m;v) = 0$, the underlying PDE. To that end, the finite element method is introduced in Sect. 1.1. There are a number of factors motivating the adoption of the finite element method in this work rather than any of the other commonly employed numerical methods for PDEs. The finite element method combines geometric flexibility, the availability of a wide range of discrete function spaces enabling practitioners to choose those with optimal properties for a given problem, and a particularly elegant mathematical representation which facilitates rigorous analysis. In particular, the finite element method makes the role of particular discrete function spaces, and their dual spaces, explicit in the discretisation. As will become apparent, the relationship between these function spaces and mesh dependence is critical. The choice of the finite element method also presents the opportunity to present the derivation of the adjoint PDE in Sect. 1.4 in the setting of a variational problem. This is in contrast to the common practice in works such as [17] of deriving the adjoint of the strong form of the PDE. It is hoped that the variational derivation presented here will be particularly useful to those who employ the finite element method. However, it is important to note that the issues considered in this work carry over to other discretisation approaches: it is not possible to avoid mesh dependence simply by switching to a finite difference or finite volume formulation.

Second, we must be able to compute derivatives of J with respect to the parameters m. Many problems arising in PDE-constrained optimisation have the common

feature that the solution of the governing PDE is computationally demanding and that a large number of parameters need to be optimised. Naïve differentiation methods scale linearly with the number of parameters, which makes them computationally too costly for many applications. However, these drawbacks can be overcome through use of the adjoint approach for computing derivatives, which is introduced in Sect. 1.4.

Finally, suitable optimisation methods must be employed. In many applications the situation arises in which the parameters m denote a function rather than a set of numbers. For example, the shape of the aerofoil in the above mentioned example can be considered as a function. Optimisation methods that are formulated on abstract spaces, including functions spaces and Euclidean spaces, are introduced in Sect. 1.5.

1.1 The Finite element method

The finite element method approximates the solution of the weak form of a PDE by replacing the infinite-dimensional spaces of solution functions and test functions with finite-dimensional subspaces. This is achieved by decomposing the underlying domain into a collection of subdomains, or cells, on each of which the admissible functions are a defined set of polynomials. Discrete integral operators are evaluated on each cell and the results recombined to form and solve the global problem over the entire domain. In this section the concept of finite elements is introduced using two simple but representative PDEs, namely, Poisson's equation and the time-dependent viscous Burgers equation. The latter is nonlinear and its numerical solution additionally requires a discretisation of the temporal dimension. The former is linear and is the classical example used when introducing finite elements. It is therefore the starting point of this section.

1.1.1 Poisson's equation

The Poisson equation,

$$-\Delta u = f \quad \text{on} \quad \Omega, \tag{1.2}$$

is the simplest elliptic partial differential equation and will serve as a prime example for the derivation of the finite element method. Here we will assume that we are in two spatial dimensions and thus

$$\Delta := \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}, \tag{1.3}$$

denotes the Laplace operator (or Laplacian) for functions on $\mathbb{R}^2$. Moreover, f is a real-valued source function on the domain $\Omega \subset \mathbb{R}^2$ and u the associated real-valued solution. The solution u also satisfies boundary conditions defined on the boundary $\partial\Omega$ of the domain. For the sake of simplicity we only introduce the two most common boundary conditions given by:

$$\text{Dirichlet-type:} \quad u = g_D \quad \text{on} \quad \partial\Omega_D, \tag{1.4}$$

$$\text{Neumann-type:} \quad \frac{\partial u}{\partial n} = g_N \quad \text{on} \quad \partial\Omega_N, \tag{1.5}$$

where $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. Furthermore, $n = (n_1, n_2)^T$ denotes the unit outward normal vector to $\partial\Omega$ and

$$\frac{\partial u}{\partial n} = \left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2} \right) \cdot (n_1, n_2)^T. \tag{1.6}$$

Thus, a Neumann boundary condition prescribes the derivative of the solution in the direction pointing directly outwards relative to the domain, whereas a Dirichlet boundary condition prescribes the value of the solution itself on the boundary. In the case where $\Omega_D = \emptyset$ we speak of pure Neumann boundary conditions, and if $\Omega_N = \emptyset$ we speak of pure Dirichlet boundary conditions. Poisson's equation 1.2 together with the boundary conditions (1.4, 1.5) is called a boundary value problem.

1.1.1.1 Derivation of weak solutions

A function u that is sufficiently smooth and satisfies (1.2) as well as (1.4, 1.5) is called a classical solution to the boundary value problem. In the case of Dirichlet boundary conditions, a classical solution u must be twice differentiable on Ω and continuous on the closure $\bar{\Omega}$, i.e. $u \in C^2(\Omega) \cap C^0(\bar{\Omega})$. It can be shown analytically that for sufficiently smooth source data f and boundary $\partial\Omega$, a classical solution exists and is unique. However, many problems lack these smoothness or regularity conditions such that the classical formalism becomes unsuitable. This can be the case for non-convex domains; see [9, p.11-13]. Further, consider the case in which f in (1.2) is a discontinuous function. Were there a classical solution u, one would have $f = \Delta u \in C^0(\Omega)$ contradicting the assumption on the discontinuous nature of f.

Discontinuous sources are physically perfectly reasonable. In thermodynamics, the Poisson equation describes the steady-state temperature profile u under some heating source f. If $f = 1$ on one part of the domain, and $f = 0$ on the rest of the domain, this corresponds to a constant heating applied to only part of the domain.

In order to overcome the limitations of classical solutions, we relax the regularity properties required of the solution by adopting an alternative formulation of the boundary value problem. To this end, we suppose that the solution $u : \Omega \to \mathbb{R}$ belongs to some suitable function space V. We multiply both sides of (1.2) by an arbitrary test function $v \in V'$ from some appropriate set of test functions and integrate

over the whole domain. The problem thus becomes, find $u \in V$ such that

$$-\int_\Omega \Delta u(x)v(x)\,\mathrm{d}x = \int_\Omega f(x)v(x)\,\mathrm{d}x \qquad \forall v \in V'. \tag{1.7}$$

Applying integration by parts to the left hand side yields

$$\int_\Omega \nabla u(x)\cdot\nabla v(x)\,\mathrm{d}x - \int_{\partial\Omega} \frac{\partial u}{\partial n}(s)v(s)\,\mathrm{d}s = \int_\Omega f(x)v(x)\,\mathrm{d}x, \tag{1.8}$$

where $\nabla := (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2})$ denotes the nabla operator. A function u satisfying (1.8) is called a weak solution since it requires less smoothness than a classical one. Indeed, a weak solution u does not need to be twice differentiable. Instead regularity conditions on the spaces V and V' required to make (1.8) well-posed are formulated in terms of the integrability of functions and their first derivatives. One important function space in this context is the space of Lebesgue square-integrable functions

$$L^2(\Omega) := \left\{ u : \Omega \to \mathbb{R} \;\middle|\; \int_\Omega u^2(x)\,\mathrm{d}x < \infty \right\}, \tag{1.9}$$

with the associated inner product

$$\langle u, v\rangle_{L^2(\Omega)} := \int_\Omega u(x)v(x)\,\mathrm{d}x, \tag{1.10}$$

and norm

$$\|u\|_{L^2(\Omega)} := \left(\langle u, u\rangle_{L^2(\Omega)}\right)^{1/2} = \sqrt{\int_\Omega u^2(x)\,\mathrm{d}x}. \tag{1.11}$$

Applying pure Neumann conditions to (1.8) yields

$$\int_\Omega \nabla u(x)\cdot\nabla v(x)\,\mathrm{d}x - \int_{\partial\Omega} g_N(s)v(s)\,\mathrm{d}s = \int_\Omega f(x)v(x)\,\mathrm{d}x. \tag{1.12}$$

Note that any solution to (1.12) is only defined up to a constant. Uniqueness holds if some of the Neumann boundary is replaced by a Dirchlet boundary.
For (1.12) to be well-defined we require every integral term to be finite. For the forcing term to be finite obviously requires $f, v \in L^2(\Omega)$. For the boundary term we additionally require that $g_N \in L^2(\partial\Omega)$ and the so called trace inequality, which relates an integral over the boundary to an integral over the whole domain; see [8, Sect. 1.5.1] for details. For the Laplacian term we note that if $\frac{\partial u}{\partial x_i}, \frac{\partial v}{\partial x_i} \in L^2(\Omega)$ for all i then

$$\left| \int_\Omega \nabla u(x) \cdot \nabla v(x) \, \mathrm{d}x \right| = \left| \sum_i \int_\Omega \frac{\partial u}{\partial x_i}(x) \frac{\partial v}{\partial x_i}(x) \, \mathrm{d}x \right| \tag{1.13}$$

$$\leq \sum_i \left| \left\langle \frac{\partial u}{\partial x_i}, \frac{\partial v}{\partial x_i} \right\rangle_{L^2(\Omega)} \right| \tag{1.14}$$

$$< \infty. \tag{1.15}$$

From these considerations, we conclude that a natural space for both the weak solution u of (1.12) as well as the associated test functions v to exist in is the Sobolev space $H^1(\Omega)$ defined by

$$H^1(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,\middle|\, u \in L^2(\Omega),\ \frac{\partial u}{\partial x_i} \in L^2(\Omega)\ \ \forall i \right\}. \tag{1.16}$$

In summary, while a classical solution of the boundary value problem (1.2) and (1.4, 1.5) for Poisson's equation must be twice differentiable, a weak solution need only be square integrable and have square-integrable first derivatives.

1.1.1.2 Weak solutions for higher-order PDEs

In a manner analogous to that presented above for Poisson's equation, higher-order partial differential equations whose classical solutions require higher-order differentiability (triple or more) can be rewritten in their weak integral form by applying integration by parts. The regularity conditions posed on the (weak) solution thereby reduce again to integrability conditions on the solution itself and derivatives of lower order than in the classical formalism. Consider for example the biharmonic equation

$$\Delta^2 u = f, \tag{1.17}$$

with $\Delta^2 = \sum_{i=1}^2 \sum_{j=1}^2 \partial_i^2 \partial_j^2$. Multiplying (1.17) by a test function, integrating, and applying integration by parts twice naturally leads to integrability conditions on the second derivatives of the solution. A natural choice for the space of solutions is the Sobolev space $H^2(\Omega)$ given by

$$H^2(\Omega) := \left\{ u : \Omega \to \mathbb{R} \,\middle|\, u \in L^2(\Omega),\ \frac{\partial u}{\partial x_i} \in L^2(\Omega),\ \frac{\partial^2 u}{\partial x_i \partial x_j} \in L^2(\Omega)\ \ \forall i, j \right\}. \tag{1.18}$$

1.1.1.3 Weak derivatives

The partial derivatives appearing in $H^1(\Omega)$ and $H^2(\Omega)$ are not understood in the usual strong sense. Recall the usual definition

$$\frac{\partial u}{\partial x_i}(x) = \lim_{\varepsilon \to 0} \frac{u(x+\varepsilon e_i) - u(x)}{\varepsilon}, \tag{1.19}$$

where e_i denotes the unit vector in the x_i direction for $i = 1,2$. Instead, in a manner analogous to the weak formulation, they are understood in a spatially averaged way with respect to a set of test functions. These derivatives are therefore called weak derivatives, and can be defined for locally integrable functions, i.e. that are integrable over compact sets inside Ω. The space of these functions is denoted by $L^1_{\text{loc}}(\Omega)$. We say that $u \in L^1_{\text{loc}}(\Omega)$ has a weak derivative with respect to x_i, denoted by $D^{x_i}_w u$, if there is a function $w_{x_i} \in L^1_{\text{loc}}(\Omega)$ such that

$$\int_\Omega w_{x_i}(x) v(x) \mathrm{d}x = -\int_\Omega u(x) \frac{\partial v}{\partial x_i}(x) \mathrm{d}x \quad \forall v \in C^\infty_0(\Omega), \tag{1.20}$$

where $C^\infty_0(\Omega)$ is the set of smooth functions on Ω with compact support. In that case, we set $D^{x_i}_w u = w_{x_i}$. Definition (1.20) can be easily generalised for higher-order partial derivatives, appearing for instance in $H^2(\Omega)$.

It follows immediately from the integration by parts formula that any differentiable function is also weakly differentiable. Weak derivatives are useful since they allow for important classes of functions to have a (weak) derivative where classically they would not. For instance, $H^1(\Omega)$ contains piecewise linear polynomials, which are an essential class of functions used for finite element approximations. We shall return to this later when finite element spaces are introduced.

1.1.1.4 Weak formulation

In order to solve the generic integral form (1.8) of the boundary value problem with boundary conditions (1.4, 1.5), let us assume that $\int_{\partial\Omega_D} \mathrm{d}s > 0$, i.e. the boundary conditions are not purely Neumann. Following [8, Sect. 1.2], we define the solution and test function spaces by

$$H^1_D := \left\{ u \in H^1(\Omega) \middle| u = g_D \text{ on } \partial\Omega_D \right\}, \tag{1.21}$$

$$H^1_0 := \left\{ v \in H^1(\Omega) \middle| v = 0 \text{ on } \partial\Omega_D \right\}, \tag{1.22}$$

respectively. The Dirichlet boundary conditions are contained in the set H^1_D of weak solutions while the test function space vanishes over the corresponding part of the boundary. The Neumann boundary conditions do not explicitly restrict the solution or the test functions.

The complete weak formulation of the boundary value problem can now be specified as follows: find $u \in H^1_D(\Omega)$ such that

$$\int_\Omega \nabla u(x) \cdot \nabla v(x) \,\mathrm{d}x - \int_{\partial\Omega_N} g_N(s) v(s) \,\mathrm{d}s = \int_\Omega f(x) v(x) \,\mathrm{d}x \quad \forall v \in H^1_0(\Omega). \tag{1.23}$$

The reader might reasonably ask what the physical interpretation of a weak solution to a PDE might be in contrast to a classical one. In the classical formalism, the PDE solution is prescribed by the equations of the PDE at every point of the domain and its boundary. However, any PDE is just a mathematical representation that approximates an actual physical phenomenon, whose observations can never be perfectly localised. It is therefore physically more meaningful, and conforms with what is actually observed in an experiment of the real world, to instead consider what happens on average over a small region of space. This is precisely what is described in the weak formulation of a PDE. This can be seen if we take a smooth test function v whose support is a small neighbourhood of some point $x_0 \in \Omega$ and whose integral is equal to one. In this case

$$\int_\Omega u(x)v(x)\mathrm{d}x \tag{1.24}$$

represents a spatial average of the values of u in a small neighbourhood of x_0. Further, the solutions to many physical problems contain discontinuities such as shock waves. The weak formulation allows for the existence of these solutions in a mathematically rigorous framework.

1.1.1.5 Galerkin finite element method

The function spaces H^1_D and H^1_0 from (1.21, 1.22) are of infinite dimension. To see this for H^1_0 suppose $\Omega = (0,\pi)^2 \subset \mathbb{R}^2$, then the set of functions $\{\lambda_k\}_{k\in\mathbb{N}}$ with

$$\lambda_k(x_1,x_2) := \sin(kx_1)\cos\left(kx_2 + \frac{\pi}{2}\right) \quad \forall (x_1,x_2) \in \Omega, \tag{1.25}$$

are linearly independent, and $\lambda_k \in H^1_0(\Omega)\ \forall k \in \mathbb{N}$. Thus, $H^1_0(\Omega)$ cannot be of finite dimension.

Since computational resources are finite, numerically solving the weak form (1.23) requires a finite-dimensional approximation. Let us therefore choose an appropriate subspace $V_{0,h} \subset H^1_0$ of test functions of dimension $d \in \mathbb{N}$, which we call a finite element space. Any $v_h \in V_{0,h}$ can be represented with respect to a set of basis functions $\{\varphi_1,\dots,\varphi_d\}$ for $V_{0,h}$, which we shall call shape functions, and coefficients $v_{h,1},\dots,v_{h,d} \in \mathbb{R}$, by

$$v_h = \sum_{i=1}^{d} v_{h,i}\varphi_i. \tag{1.26}$$

The Dirichlet boundary condition in the solution is interpolated by an additional set of shape functions $\{\varphi_{d+1},\dots,\varphi_{d+d_D}\}$ and associated coefficients $g_{h,i}$. Thus, the finite element solution $u_h \in V_{D,h}$, where $V_{D,h} \subset H^1_D$ is spanned by the set of basis functions $\{\varphi_{d+1},\dots,\varphi_{d+d_D},\varphi_{d+1},\dots,\varphi_{d+d_D}\}$, can be expressed in a unique way via the coefficients $u_{h,1},\dots,u_{h,d}$ and $g_{h,d+1},\dots g_{h,d+d_D}$ by

$$u_h = \sum_{i=1}^{d} u_{h,i}\varphi_i + \sum_{i=d+1}^{d+d_D} g_{h,i}\varphi_i. \tag{1.27}$$

Replacing $u \in H_D^1(\Omega)$ and $v \in H_0^1(\Omega)$ in the weak formulation (1.23) by $u_h \in V_{D,h}$ and $v_h \in V_{0,h}$, respectively, yields the finite-dimensional approximation we are looking for, which is commonly termed the Galerkin finite element approximation, i.e. find $u_h \in V_{D,h}$ such that

$$\int_\Omega \nabla u_h(x) \cdot \nabla v_h(x)\,\mathrm{d}x - \int_{\partial\Omega_N} g_N(s) v_h(s)\,\mathrm{d}s = \int_\Omega f(x) v_h(x)\,\mathrm{d}x \quad \forall v_h \in V_{0,h}. \tag{1.28}$$

Making use of the representations (1.26) and (1.27) of v_h and u_h with respect to their basis functions, respectively, (1.28) is equivalent to

$$\begin{aligned} \sum_{j=1}^{d} u_{h,j} \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i\,\mathrm{d}x = \int_\Omega \varphi_i f\,\mathrm{d}x + \int_{\partial\Omega_N} \varphi_i g_N\,\mathrm{d}s \\ - \sum_{j=d+1}^{d+d_D} u_{h,j} \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i\,\mathrm{d}x \quad \forall i = 1,\ldots,d. \end{aligned} \tag{1.29}$$

This system of equations can be rewritten in matrix form, which we term the Galerkin system and is given by

$$\mathrm{A}\mathbf{u} = \mathbf{f}, \tag{1.30}$$

where $\mathbf{u} := (u_{h,1},\ldots,u_{h,d})^T$, the entry A_{ij} of the ith row and jth column of A is given by

$$A_{ij} = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i\,\mathrm{d}x, \tag{1.31}$$

and the ith component f_i of the vector $\mathbf{f}$ is given by

$$f_i = \int_\Omega \varphi_i f\,\mathrm{d}x + \int_{\partial\Omega_N} \varphi_i g_N\,\mathrm{d}s - \sum_{j=d+1}^{d+d_D} u_{h,j} \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j\,\mathrm{d}x. \tag{1.32}$$

The matrix A is usually referred to as the *stiffness matrix*, while $\mathbf{f}$ is often termed the *load vector*.

Spatially, the domain $\Omega \subset \mathbb{R}^2$ is divided into a mesh of polygons such as triangles or quadrilaterals, or indeed other two-dimensional shapes; these are termed the elements or cells. The finite element spaces are chosen such that the basis functions $\varphi_1,\ldots,\varphi_d$ have compact support, i.e. they are each only locally nonzero on the mesh, thereby creating local matrix and vector contributions which are in general far smaller than A and $\mathbf{f}$. The local matrix and vector quantities can easily be computed

and can then be assembled into the global Galerkin system that models the entire problem.

1.1.1.6 Triangular Lagrange finite elements

For the sake of simplicity and following [8, Sect. 1.3.1], let us assume that the domain Ω is polygonal, i.e. Ω can be tessellated entirely with triangles $\Delta_k \subset \Omega$ for $k = 1, \ldots, K$, i.e.

$$\cup_{k=1}^{K} \bar{\Delta}_k = \bar{\Omega}, \tag{1.33}$$

$$\Delta_k \cap \Delta_\ell = \emptyset \quad \forall k \neq \ell. \tag{1.34}$$

We shall call the set $\{\Delta_1, \ldots, \Delta_K\}$ a triangulation and denote it by $\mathscr{T}_h$. The edges and vertices of the set of triangles form a mesh, and we shall refer to each triangle as a cell or an element. The finite element spaces we shall consider are the continuous piecewise polynomials of degree p. This means that the space restricted to any cell is the space of polynomials of degree p on that cell, and with the space constrained to be continuous along the interior edges and at every vertex of the mesh. The natural basis functions for this space are polynomials of degree p which take the value 1 at a particular nodal point and vanish at the nodal points for all other basis functions. The simplest choice for such basis functions is the set $\mathrm{P}_1(\mathscr{T}_h)$ of piecewise linear polynomials. Any $\varphi_j \in \mathrm{P}_1(\mathscr{T}_h)$ is a linear function on each of the surrounding elements of vertex j and is identically equal to zero on all other elements. Figure 1.1 shows an example of a P_1 basis function which is nonzero on six elements of a triangular mesh. Clearly, φ_j is not differentiable (at vertex j and along the border of the patch of its surrounding elements). However, it is simple to show that φ_j is weakly differentiable and therefore that $\mathrm{P}_1(\mathscr{T}_h) \subset H^1(\Omega)$.

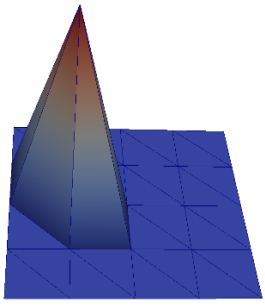
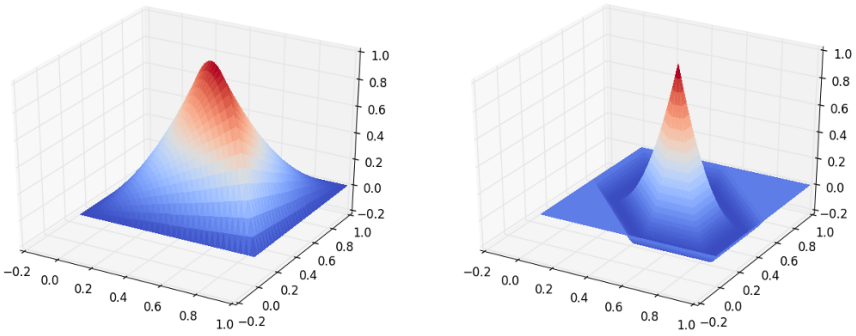


Fig. 1.1: P_1 basis function

The definition of the set $\mathrm{P}_2(\mathscr{T}_h)$ of piecewise quadratic functions requires additional nodes located at the mid-point of each mesh edge. As with P_1, a basis function $\varphi_j \in \mathrm{P}_2 \subset H^1(\Omega)$ is nonzero only on the elements surrounding node j. More pre-

cisely, any $\varphi_j \in \mathrm{P}_2(\mathscr{T}_h)$ is a quadratic function on each of the elements adjacent to node j and is identically equal to zero on all other elements. Further, φ_j is equal to one at node j and equal to zero on all other nodes in the mesh. Figure 1.2 shows the two types of P_2 basis functions, the first one being equal to 1 on an edge mid-point of an element and the second one being equal to 1 on a vertex. Finite elements can be defined in a similar manner for piecewise polynomial functions of arbitrary order $p \in \mathbb{N}$.



(a) being equal to 1 on the common edge node of two elements

(b) being equal to 1 on the common vertex node of six elements

Fig. 1.2: P_2 basis functions

The Galerkin matrix A created by this choice of basis functions is sparse. For example, on a perfectly regular mesh, each interior vertex is surrounded by six triangles, so the matrix rows associated with the P_1 nodes at each interior vertex have seven non-zeros. More generally the number of non-zeros on each row depends only on the local mesh topology: it does not increase as the mesh is refined or the domain extended. Similarly, the rows corresponding to the interior vertex nodes of P_2 on a completely regular mesh have 19 nonzeros, while the rows associated with edge nodes have 9 nonzeros. The sparsity of A is exploited to solve the Galerkin system (1.30) efficiently. A discussion of efficient sparse linear solvers however is beyond the scope of this work, but the interested reader is referred to [6], [8] and [32] for details.

1.1.1.7 Assembly of the Galerkin system

Having created a triangulation of the domain Ω with an associated set of basis functions $\varphi_1, \ldots, \varphi_d$, the linearity of integration can be exploited to decompose the assembly of the Galerkin system (1.30) into a local integral on each cell and boundary facet. For example, we can rewrite the global stiffness matrix (1.31) as

$$\begin{aligned} A_{ij} &= \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \,\mathrm{d}x, \\ &= \sum_{\Delta_k \in \mathscr{T}_h} \int_{\Delta_k} \nabla\varphi_j \cdot \nabla\varphi_i \,\mathrm{d}x. \end{aligned} \tag{1.35}$$

On every element Δ_k there are only a small number $n_k \in \mathbb{N}$ of basis functions which are nonzero. For instance a P_1 space has only three such basis functions per element while for P_2, there are six. For each cell k, we can define a function, $s_k(\hat{\imath})$, which maps the local basis function index $\hat{\imath}$ to the corresponding global index i . From this we can define the element scatter matrix $\mathrm{S}_k[i,j] = \delta(s_k(\hat{\imath}), i)$, and hence write the global stiffness matrix as a sum of local contributions:

$$\mathrm{A} = \sum_k \mathrm{S}_k \mathrm{A}_k \mathrm{S}_K^T, \tag{1.36}$$

with the local stiffness matrix on cell k given by

$$\mathrm{A}_k[\hat{\imath},\hat{\jmath}] = \int_{\Delta_k} \nabla\varphi_{s_k(\hat{\imath})} \cdot \nabla\varphi_{s_k(\hat{\jmath})} \,\mathrm{d}x. \tag{1.37}$$

Note that the local stiffness matrix is a dense $n_k \times n_k$ matrix which contributes a sparse update to A with sparsity given by the element scatter matrix. In practice, the element scatter matrix is not actually constructed. Instead, the scatter function s_k is employed directly to add contributions to the appropriate entries of A. Any essential boundary conditions are neglected in the local stiffness matrix. They are instead applied by modifying the global stiffness matrix A, either during or after the addition of the local contributions. This process applies *mutatis mutandis* to the assembly of the global load vector from local contributions.

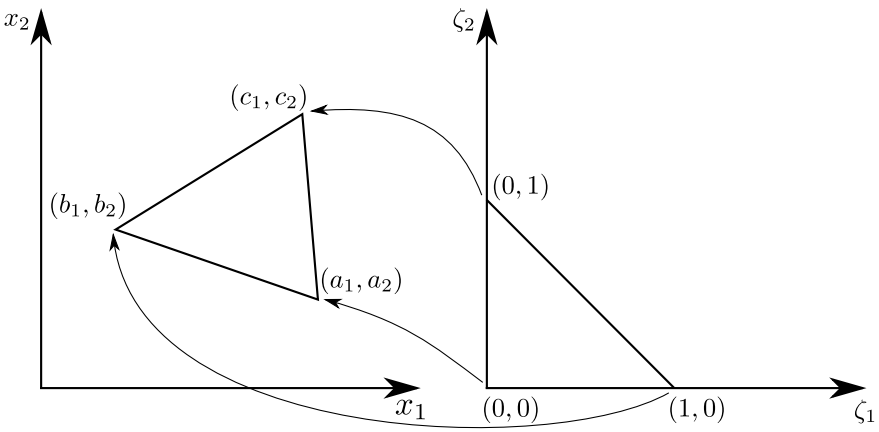


Fig. 1.3: Diffeomorphic mapping ρ_k for the P_1 element Δ^k

A convenient mechanism for the construction of the local stiffness matrix is to evaluate all element integrals on a single reference element Δ^* which is tied to each global element Δ_k by a diffeomorphic mapping, ρ_k. Here, Δ^* is formed by the vertices $(0,0),(0,1)$ and $(1,0)$. As illustrated by Fig. 1.3, Δ^* is mapped onto the local element Δ_k with vertices $a=(a_1,a_2), b=(b_1,b_2)$ and $c=(c_1,c_2)$ with ρ_k defined by,

$$\begin{aligned}\rho_k(\zeta_1,\zeta_2) &= a\tau_1(\zeta_1,\zeta_2)+b\tau_2(\zeta_1,\zeta_2)+c\tau_3(\zeta_1,\zeta_2) &(1.38)\\ &= (x_1,x_2), &(1.39)\end{aligned}$$

where:

$$\begin{aligned}\tau_1(\zeta_1,\zeta_2) &= 1-\zeta_1-\zeta_2, &(1.40)\\ \tau_2(\zeta_1,\zeta_2) &= \zeta_1, &(1.41)\\ \tau_3(\zeta_1,\zeta_2) &= \zeta_2. &(1.42)\end{aligned}$$

This enables us to define a set of n_k local basis functions for our finite element space over Δ^*: $\{\psi_i\}$ with the property that $\psi_i(\zeta)=\varphi_{s_k(i)}(\rho_k(\zeta))$, where $\zeta=(\zeta_1,\zeta_2)$. For clarity, the associated $\varphi_{s_k(i)}$ are also called global basis functions. The local basis functions enable us to pull back the local stiffness contributions to the reference space:

$$\int_{\Delta_k}\nabla\varphi_{s_k(i)}\cdot\nabla\varphi_{s_k(j)}\,\mathrm{d}x=\int_{\Delta^*}\left(J_k^T\nabla_\zeta\psi_i\right)\cdot\left(J_k^T\nabla_\zeta\psi_j\right)|J|\,\mathrm{d}\zeta, \tag{1.43}$$

where $\nabla_\zeta:=\left(\frac{\partial}{\partial\zeta_1},\frac{\partial}{\partial\zeta_2}\right)^T$ is the gradient operator in local coordinates and $J:=\nabla_\zeta\rho$ is the cell Jacobian[1]:

$$\begin{aligned}J_{\alpha,\beta} &= \frac{\partial x_\alpha}{\partial\zeta_\beta}\\ &= a_\alpha\frac{\partial\tau_1}{\partial\zeta_\beta}+b_\alpha\frac{\partial\tau_2}{\partial\zeta_\beta}+c_\alpha\frac{\partial\tau_3}{\partial\zeta_\beta}.\end{aligned} \tag{1.44}$$

The identity $\nabla=J^T\nabla_\zeta$ follows immediately by the chain rule. The details of the assembly process are beyond the scope of this work, however the interested reader is referred to [2, Sect. 0.6], [8, Sect. 1.4], and [23, Chap. 6].

1.1.1.8 Example problem

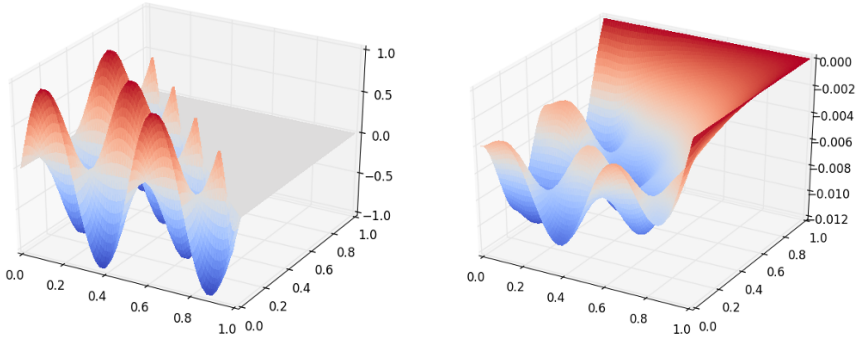
Let us consider the boundary value problem (1.2, 1.4, 1.5) for Poisson's equation with parameters

[1] Note that the Jacobian, J, is a totally separate concept from the functional to be optimised from (1.1), which is also written J.

$$\Omega = [0,1]^2, \tag{1.45}$$
$$\partial\Omega_D = \{x_1 = 1\} \cup \{x_2 = 1\}, \tag{1.46}$$
$$\partial\Omega_N = \Omega \setminus \partial\Omega_D, \tag{1.47}$$
$$g_D = 0 \quad \text{on} \quad \partial\Omega_D, \tag{1.48}$$
$$g_N = 0 \quad \text{on} \quad \partial\Omega_N, \tag{1.49}$$
$$f(x_1,x_2) = \chi_{\{x_1+x_2<1\}} \sin(4\pi x_1)\cos(4\pi x_2) \quad \forall (x_1,x_2) \in \Omega. \tag{1.50}$$

A plot of the source function f is displayed in Fig. 1.4a. Since f is discontinuous, there is no classical solution to this problem. However, it can be proved that there is a weak solution u and that u is unique, which is a direct consequence of Lax-Milgram's theorem. For details, the interested reader is referred to [2, Sect. 2.7].

An approximate solution u_h to the weak boundary value problem (1.23) with parameters (1.45−1.50) is computed numerically using the finite element method embedded within the Firedrake framework [31]. Firedrake is an automated system for the portable solution of PDEs using finite elements. It allows for a succinct implementation for a large variety of discretisations and PDEs. For an introductory tutorial to the usage of Firedrake, we refer to [11]. The code used to generate the corresponding solution in Python can be found in [33]. Here, the weak solution u_h is approximated using P_1 finite elements. A plot of u_h is displayed in Fig. 1.4b.



(a) Source f (b) Solution u_h

Fig. 1.4: Plots of the discontinuous source function f and the finite element solution u_h to the weak boundary value problem (1.23) for Poisson's equation with parameters (1.45−1.50)

1.1.2 Burgers' equation

The viscous form of Burgers' equation

$$\frac{\partial u}{\partial t} + (u \cdot \nabla) u - \eta \Delta u = 0 \quad \text{on} \quad [0,T] \times \Omega, \tag{1.51}$$

is a fundamental nonlinear partial differential equation describing the advection and diffusion of u, a velocity field. Here, $\eta > 0$ denotes an assumed to be constant viscosity or diffusion coefficient and $u : [0,T] \times \Omega \to \mathbb{R}^2$ the associated solution field over the time interval $[0,T]$, $T > 0$, and the domain $\Omega \subset \mathbb{R}^2$. Further,

$$\Delta u := (\Delta u_1,\ \Delta u_2)^T, \tag{1.52}$$

denotes the vector Laplacian of $u := (u_1, u_2)^T$. By simple vector computations, one finds

$$(u \cdot \nabla) u = \left(u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2},\ u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} \right)^T. \tag{1.53}$$

The solution of (1.51) is assumed here to satisfy Neumann boundary conditions given by

$$(n \cdot \nabla) u = 0 \quad \text{on} \quad [0,T] \times \partial\Omega, \tag{1.54}$$

for which the vector term can simply be written as

$$(n \cdot \nabla) u = \left(\frac{\partial u_1}{\partial n}, \frac{\partial u_2}{\partial n} \right)^T. \tag{1.55}$$

Since the problem at hand is time dependent, an initial condition, i.e. a condition for the state of the system at time $t = 0$, is given by

$$u(0,x) = u_0(x), \tag{1.56}$$

for a given function $u_0 : \Omega \to \mathbb{R}^2$. Combining Burgers' equation (1.51) with the initial condition (1.56) forms a so called initial value problem. Together with the boundary condition (1.54), it is also a boundary value problem.

1.1.2.1 Weak formulation

Using vector multiplication to multiply both sides of (1.51) by a test function $v : \Omega \to \mathbb{R}^2$ from some appropriate test function space and integrating over the whole domain yields

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v \,\mathrm{d}x + \int_\Omega (u \cdot \nabla) u \cdot v \,\mathrm{d}x - \eta \int_\Omega \Delta u \cdot v \,\mathrm{d}x = 0. \tag{1.57}$$

Applying the integration-by-parts formula to the Laplace term yields

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v \,\mathrm{d}x + \int_\Omega (u \cdot \nabla) u \cdot v \,\mathrm{d}x + \eta \int_\Omega \nabla u : \nabla v \,\mathrm{d}x = 0, \tag{1.58}$$

where

$$\nabla u : \nabla v := \nabla u_1 \cdot \nabla v_1 + \nabla u_2 \cdot \nabla v_2. \tag{1.59}$$

The boundary term arising from integrating by parts has disappeared here due to the application of the Neumann boundary condition (1.54).

1.1.2.2 Time discretisation

Since we are dealing with a time dependent problem, we need to discretise (1.162) in time. For simplicity and stability, the backward Euler discretisation is used. For a given number of time steps N and associated time step size $dt = T/N$, we solve

$$\begin{aligned}\int_\Omega \frac{u^{n+1}-u^n}{dt}\cdot v\,\mathrm{d}x + \int_\Omega \left(u^{n+1}\cdot\nabla\right)u^{n+1}\cdot v\,\mathrm{d}x \\ + \eta\int_\Omega \nabla u^{n+1} : \nabla v\,\mathrm{d}x = 0 \quad \forall n = 0,\ldots,N-1,\end{aligned} \tag{1.60}$$

where $u^n = u(n\cdot dt,\cdot)$ and thus $u^0 = u(0,\cdot) = u_0$. Note that every u^n is now a time independent function, i.e. $u^n : \Omega \to \mathbb{R}^2\ \forall n = 0,\ldots,N-1$. The natural space for the solution and test functions in (1.60) is the tensor product Sobolev space $H^1(\Omega)^2 := H^1(\Omega)\otimes H^1(\Omega)$, i.e.

$$H^1(\Omega)^2 = \left\{ u : \Omega \to \mathbb{R}^2 \middle| u_i, \frac{\partial u_i}{\partial x_j} \in L^2(\Omega)\ \forall i,j \right\}. \tag{1.61}$$

The complete weak formulation of the Burgers' initial value problem can now be written as follows: find $u^{n+1} \in H^1(\Omega)^2$ such that

$$\begin{aligned}\int_\Omega \frac{u^{n+1}-u^n}{dt}\cdot v\,\mathrm{d}x + \int_\Omega \left(u^{n+1}\cdot\nabla\right)u^{n+1}\cdot v\,\mathrm{d}x \\ + \eta\int_\Omega \nabla u^{n+1} : \nabla v\,\mathrm{d}x = 0 \quad \forall v \in H^1(\Omega)^2\ \forall n = 0,\ldots,N-1.\end{aligned} \tag{1.62}$$

1.1.2.3 Galerkin approximation

Consider a finite-dimensional subspace $V_h \subset H^1(\Omega)^2$. Any $u_h \in V_h$ can be represented as

$$u_h = \sum_{i=1}^{d} u_{h,i}\varphi_i. \tag{1.63}$$

In contrast to the previous, scalar, Poisson equation, the solution to Burgers' equation is vector-valued. We can construct a suitable finite element space by taking the tensor product of two scalar valued finite element spaces. Suppose $W_h \subset H^1(\Omega)$ is some such space with basis $\omega_1, \ldots, \omega_d$. Then a basis for $V_h = W_h \otimes W_h$ is given by:

$$\{e_1 \omega_i \quad 0 < i \leq d\} \quad \bigcup \quad \{e_2 \omega_i \quad 0 < i \leq d\}, \tag{1.64}$$

where e_1 and e_2 are the unit vectors in the x_1 and x_2 directions respectively.

The Galerkin approximation of Burgers' equation is a finite-dimensional approximation of the weak formulation (1.62), which can be expressed as follows: Find $u_h^{n+1} \in V_h$ such that

$$\int_\Omega \frac{u_h^{n+1} - u_h^n}{dt} \cdot v_h \, \mathrm{d}x + \int_\Omega \left(u_h^{n+1} \cdot \nabla\right) u_h^{n+1} \cdot v_h \, \mathrm{d}x + \eta \int_\Omega \nabla u_h^{n+1} : \nabla v_h \, \mathrm{d}x = 0 \qquad \forall v_h \in V_h \; \forall n = 0, \ldots, N-1. \tag{1.65}$$

1.1.2.4 Residual form

Unlike the Poisson problem, the weak formulation (1.62) of Burgers' initial value problem is nonlinear in the solution u. As a consequence, the discretised problem does not simply result in a matrix system which can be solved using linear algebra techniques. Instead, Newton-like nonlinear solvers are typically employed. The first step is to express the system in what is termed *residual form*. For the weak form of Burgers' equation (1.65) this is: find u_h^{n+1} such that

$$F(u_h^{n+1}; v_h) = 0 \quad \forall v_h \in V_h, \;\; \forall n = 0, \ldots, N-1, \tag{1.66}$$

where the *residual*, F, is defined by

$$F(u_h^{n+1}; v_h) = \int_\Omega \frac{u_h^{n+1} - u_h^n}{dt} \cdot v_h \, \mathrm{d}x + \int_\Omega \left(u_h^{n+1} \cdot \nabla\right) u_h^{n+1} \cdot v_h \, \mathrm{d}x \tag{1.67}$$

$$+ \eta \int_\Omega \nabla u_h^{n+1} : \nabla v_h \, \mathrm{d}x. \tag{1.68}$$

The semicolon indicates that F is linear in the argument to the right of the semicolon. Note that F is not linear in the first argument. Since the residual is linear in v_h, it is sufficient that (1.66) holds for every test function in the basis of V_h. That is, the problem reduces to finding $u_h^{n+1} \in V_h$ such that

$$F(u_h^{n+1}; \varphi_j) = 0 \quad \forall j = 1, \ldots, d, \;\; \forall n = 0, \ldots, N-1. \tag{1.69}$$

1.1.2.5 Linearisation

The residual form (1.66) can be employed to formulate the linearisation of the problem as required by Newton-like methods.

The linearisation of the residual F requires its differentiation with respect to its first argument u_h^{n+1}. For the sake of simplicity we shall omit the upper 'n' index for the time discretisation in the following, i.e. u_h^{n+1} is replaced by u_h. Since at every time step $n = 0, \ldots, N-1$ an equation with equivalent structure is solved, there is no loss of generality in what follows. Since u_h is not an element of a Euclidean space but is a function, a more general notion of differentiability than for functions on Euclidean spaces is needed. The Gâteaux derivative of $F(\cdot\,; v_h)$ at u_h is given by

$$\mathrm{d}F_{u_h}(u_h; v_h, \hat{u}) = \lim_{\varepsilon \to 0} \frac{F(u_h + \varepsilon \hat{u}; v_h) - F(u_h; v_h)}{\varepsilon} \quad \forall \hat{u} \in V_h. \tag{1.70}$$

In other words, $\mathrm{d}F_{u_h}(u_h; v_h, \hat{u})$ denotes the directional derivative of $F(\cdot\,; v_h)$ with respect to u_h, at the current value of u_h and in the "direction" $\hat{u}$. The subscript will be omitted when there is no ambiguity about the variable with respect to which the derivative is being taken. A more formal introduction to Gâteaux derivatives is given in Sect. 1.2.

The Gâteaux derivative of the Burgers' residual can be explicitly computed by

$$\begin{aligned}
\mathrm{d}F(u_h; v_h, \hat{u}) = \lim_{\varepsilon \to 0} \Bigg[& \int_\Omega \left(\frac{\hat{u}}{dt} + (\hat{u} \cdot \nabla) u_h + ((u_h + \varepsilon \hat{u}) \cdot \nabla) \hat{u} \right) \cdot v_h && (1.71) \\
& + \eta \nabla \hat{u} : \nabla v_h \, \mathrm{d}x \Bigg] && (1.72) \\
= & \int_\Omega \left(\frac{\hat{u}}{dt} + (\hat{u} \cdot \nabla) u_h + (u_h \cdot \nabla) \hat{u} \right) \cdot v_h + \eta \nabla \hat{u} : \nabla v_h \, \mathrm{d}x. && (1.73)
\end{aligned}$$

It is easy to see that $\mathrm{d}F_{u_h}$ is linear in both v_h and $\hat{u}$. This linearity will be used to solve the residual form of Burgers' initial value problem.

1.1.2.6 Solving the residual form

Since the derivative of the residual is available, a straightforward way to solve the residual form (1.66) is to apply Newton's method. Just as in the Euclidean case, Newton's method on function spaces consists of approximating the function, in this case the residual, by its Taylor expansion only up to the linear term and solving for the update that will set the approximation to zero. The linear Taylor series approximation of $F(\cdot\,; v_h)$ at a perturbed solution $u_h + \hat{u}$ is given by

$$F(u_h + \hat{u}; v_h) \approx F(u_h; v_h) + \mathrm{d}F(u_h; v_h, \hat{u}) \quad \forall v_h \in V_h. \tag{1.74}$$

Thus, if we write the $(k+1)$th Newton iterate u_h^{k+1} as

$$u_h^{k+1} = u_h^k + \hat{u}, \tag{1.75}$$

then the update $\hat{u} \in V_h$ is the solution to

$$\mathrm{d}F(u_h^k; v_h, \hat{u}) = -F(u_h^k; v_h) \quad \forall v_h \in V_h. \tag{1.76}$$

Due to the linearity of $\mathrm{d}F(u_h^k; v_h, \hat{u})$ in $\hat{u}$, Eq. (1.76) is simply a linear finite element problem. More precisely, using the linearity of $\mathrm{d}F(u_h^k; v_h, \hat{u})$ in $\hat{u}$ and the linear independence of different basis functions in V_h this can be written as

$$\sum_{j=1}^{d} \mathrm{d}F(u_h^k; \varphi_i, \varphi_j)\hat{u}_j = -F(u_h^k; \varphi_i) \quad \forall i = 1, \ldots, d, \tag{1.77}$$

where $\hat{u}_1, \ldots, \hat{u}_d$ are the coefficients of the expansion of $\hat{u}$ in terms of the basis functions. In matrix form, the last equation can be written as

$$\hat{\mathrm{A}}\hat{\mathbf{u}} = \hat{\mathbf{f}}, \tag{1.78}$$

where the entry in the ith row and jth column of $\hat{A}$ is given by

$$\hat{A}_{ij} = \mathrm{d}F(u_h^k; \varphi_i, \varphi_j) \tag{1.79}$$

$$= \int_\Omega \left(\frac{\varphi_j}{dt} + (\varphi_j \cdot \nabla)\, u_h^k + \left(u_h^k \cdot \nabla \right) \varphi_j \right) \cdot \varphi_i + \eta \left(\nabla \varphi_j : \nabla \varphi_i \right) \mathrm{d}x \quad \forall i, j = 1, \ldots, d, \tag{1.80}$$

$\hat{\mathbf{u}} = (\hat{u}_1, \ldots, \hat{u}_d)$, and $\hat{\mathbf{f}} = (\hat{f}_1, \ldots, \hat{f}_d)$ with

$$\hat{f}_i = -F(u_h^k; \varphi_i) \quad \forall i = 1, \ldots, d. \tag{1.81}$$

As was the case for Poisson's equation (1.30), this is a linear system which can be solved using standard linear algebra techniqes. In essence, Newton's method converts the nonlinear system into a (hopefully convergent) series of linear problems. More sophisticated nonlinear solvers such as line search Newton methods are available, but the core process of linearising the equations to produce a sequence of linear finite element problems is common to these approaches.

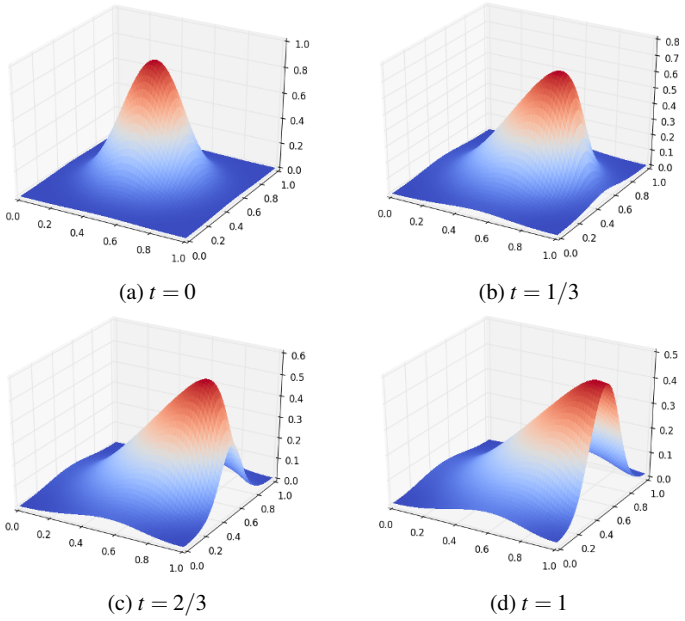
Under certain conditions, Newton's method is guaranteed to generate a sequence $(u_h^k)_k$ of approximations that converges to the correct solution of the original nonlinear problem. However, we still need to decide when to stop the algorithm and accept the most recent update as a solution. This requires more involved consideration of the appropriate norms to employ. We therefore postpone its consideration to Sect. 1.3.3.

1.1.2.7 An example problem

Let us now consider the Burgers' initial value problem (1.51, 1.54, 1.56) with parameters

$$\Omega = [0,1]^2, \tag{1.82}$$
$$T = 1, \tag{1.83}$$
$$\eta = 0.01, \tag{1.84}$$
$$u_0(x_1,x_2) = \left(\exp\left[-20\cdot\left(\left(x_1-\frac{1}{2}\right)^2+\left(x_2-\frac{1}{2}\right)^2\right)\right],\ 0\right) \quad \forall (x_1,x_2)\in\Omega. \tag{1.85}$$

An approximate solution u_h to the weak formulation (1.62) of Burgers' initial value problem with parameters (1.82−1.85) is computed numerically using finite elements in Firedrake. The code that was used to generate the corresponding solution in Python can be found in [33]. Here, the vector-valued weak solution u_h is approximated using P_2 vector-valued finite elements. A plot of the first vector component u_{1_h} of u_h at successive points in time is displayed in Fig. 1.5.



(a) $t = 0$ (b) $t = 1/3$

(c) $t = 2/3$ (d) $t = 1$

Fig. 1.5: Plots of the first vector component u_{1_h} of the finite element solution u_h to the weak Burgers' initial value problem (1.62) with parameters (1.82−1.85) at successive points in time

1.2 Hilbert spaces

The structure of a Hilbert space arises naturally in the finite element method since the weak formulation of a PDE involves an inner product with the test functions. One of the key properties of a Hilbert space is its close relationship with its dual space. The key result characterising this relationship is the Riesz representation theorem.

A Hilbert space is a complete vector space equipped with an inner product, $\langle \cdot, \cdot \rangle$. The availability of the inner product makes it possible to measure the angles between, and the length of, vectors in a Hilbert space. Completeness is the property that any Cauchy sequence in a Hilbert space converges to a limit in that space. This ensures that there are no "missing" limits with the consequence that the usual techniques of limit calculus can be applied. Here, we only consider Hilbert spaces over the real field, but most of the results here can be generalised to Hilbert spaces over other (for example complex) fields.

The inner product on a Hilbert space induces a norm defined for any $u \in H$ by:

$$\|u\|_H = \sqrt{\langle u, u \rangle}. \tag{1.86}$$

1.2.1 Dual spaces and the Riesz representation theorem

Let $F : H \to \mathbb{R}$ be a linear functional on a Hilbert space H. We call F continuous if

$$\exists c > 0 : |F(u)| \leq c\|u\|_H \quad \forall u \in H. \tag{1.87}$$

The set $\mathscr{L}(H, \mathbb{R})$ of continuous linear functionals on H is called the dual space of H and is also denoted by H^*, i.e.

$$H^* := \{F : H \to \mathbb{R} \mid F \text{ is linear and continuous}\}. \tag{1.88}$$

The Hilbert space H induces on its dual space H^* an operator norm $\|\cdot\|_{H^*}$ defined by:

$$\|F\|_{H^*} := \sup_{\|u\|_H = 1} |F(u)| \quad \forall F \in H^*. \tag{1.89}$$

The one fundamental relationship between a Hilbert space H and its dual space H^* is that these spaces are isometrically isomorphic to each other. In more comprehensible terms, this property is expressed in the following theorem.

Theorem 1.1 (The Riesz representation theorem). *Let H be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_H$. For any $F \in H^*$ there is a unique $u_F \in H$ such that*

$$F(v) = \langle u_F, v \rangle_H \quad \forall v \in H,$$

and that $\|F\|_{H^*} = \|u_F\|_H$. Conversely, for any $u \in H$ the functional F_u defined by

$$F_u(v) := \langle u, v\rangle_H \quad \forall v \in H, \tag{1.90}$$

is an element of H^*.

The map $\mathscr{R}_H : H^* \to H$ that sends F to its unique representative $u_F \in H$ is called the Riesz map. Further, $\mathscr{R}_H(F) = u_F$ is called the Riesz representer of F in H.

In simple terms, the theorem sates that any element of the dual space H^* can be uniquely represented by an element in the primal space H and vice versa, while their respective norms are equivalent.

The Riesz representation theorem is of essential importance for this work and will be used repeatedly. It is therefore appropriate to provide a proof, which follows [1, Sect. 6.1].

Proof (Riesz representation theorem). It is enough to show that the inverse mapping of the Riesz map $\tau := \mathscr{R}_H^{-1}$ given by

$$\tau(u)(v) \mapsto \langle u, v\rangle_H \quad \forall u, v \in H, \tag{1.91}$$

is an isometric isomorphism. Let $u \in H$. Obviously, $\tau(u)$ is linear. The Cauchy-Schwarz inequality yields

$$|\tau(u)(v)| \leq \|u\|_H \cdot \|v\|_H \quad \forall v \in H. \tag{1.92}$$

Therefore, $\tau(u)$ is continuous, which means $\tau(u) \in H^*$, and $\|\tau(u)\|_{H^*} \leq \|u\|_H$ by definition of the operator norm. But since $\|\tau(u)(u)\| = \|u\|^2$, we also have $\|\tau(u)\|_{H^*} \geq \|u\|_H$. Hence, τ is an isometry. Let $u_0 \in H$ such that $\tau(u_0) = 0 \in H^*$. Then,

$$0 = \|\tau(u_0)\|_{H^*} = \|u_0\|_H, \tag{1.93}$$

and thus $u_0 = 0 \in H$. Hence, τ is injective due to its linearity in u. It remains to show that τ is surjective. Consider $0 \neq F_0 \in H^*$ and let P be the orthogonal projection from H onto the closed null space

$$\mathscr{N}(F_0) := \{u \in H : F_0(u) = 0\}. \tag{1.94}$$

The existence and uniqueness of P is ensured by the projection theorem [1, Sect. 4.3]. Let $e \in H$ such that $F_0(e) = 1$, and define

$$u_0 := e - Pe. \tag{1.95}$$

Hence, $F_0(u_0) = F_0(e) - F_0(Pe) = 1$, which implies $u_0 \neq 0$. Using the characteristic property of the projection, i.e. $e - Pe \in \mathscr{N}(F_0)^\perp$, we have

$$\langle u_0, v\rangle_H = 0 \quad \forall v \in \mathscr{N}(F_0). \tag{1.96}$$

As $F_0(u - F_0(u)u_0) = F_0(u) - F_0(u) \cdot 1 = 0$, (1.96) implies that

$$\langle u_0, u\rangle_H = \langle u_0, u - F_0(u)u_0\rangle_H + \langle u_0, F_0(u)u_0\rangle_H \tag{1.97}$$
$$= \langle u_0, F_0(u)u_0\rangle_H \tag{1.98}$$
$$= F_0(u)\|u_0\|^2 \quad \forall u \in H. \tag{1.99}$$

Dividing by $\|u_0\|^2$ yields

$$F_0(u) = \left\langle \frac{u_0}{\|u_0\|^2}, u \right\rangle_H = \tau\left(\frac{u_0}{\|u_0\|^2}\right)(u) \quad \forall u \in H, \tag{1.100}$$

which concludes the proof.

1.2.2 Fréchet derivatives

Fréchet differentiability is a generalisation of the differentiability from functions on Euclidean space to more general spaces such as Hilbert spaces. Using the Fréchet derivative it is possible to extend core results from Euclidean analysis such as Taylor's theorem to a more general setting.

Let $F : U \subset H \to \mathbb{R}$ be a functional on an open set $U \neq \emptyset$ of some Hilbert space H. We call F directionally differentiable if

$$\mathrm{d}F(u;h) := \lim_{\varepsilon \to 0^+} \frac{F(u+\varepsilon h) - F(u)}{\varepsilon}, \tag{1.101}$$

exists for all $u, h \in U$. We call F Gâteaux-differentiable if F is directionally differentiable and the directional derivative $\mathrm{d}F(u;\cdot) : h \mapsto \mathrm{d}F(u;h)$ is continuous and linear , i.e. $\mathrm{d}F(u;\cdot) \in H^* \ \forall u \in U$. Further, we call F Fréchet-differentiable if F is Gâteaux-differentiable and if

$$\lim_{\|h\|_H \to 0} \frac{\|F(u+h) - F(u) - \mathrm{d}F(u;h)\|}{\|h\|_H} = 0 \quad \forall u \in U. \tag{1.102}$$

Since $\mathrm{d}F(u;\cdot) \in H^*$,

$$\mathrm{d}F(u;v) = \langle \mathscr{R}(F'(u)), v\rangle_H \quad \forall v \in U, \tag{1.103}$$

where $\mathscr{R}(\mathrm{d}F(u,\cdot))$ denotes the Riesz representer of $\mathrm{d}F(u;\cdot)$ in H. That is, the Fréchet derivative of a functional on H can be represented with respect to an element of H.

If F is Fréchet-differentiable and $\mathrm{d}F(u;\cdot) \in \mathscr{L}(U,\mathbb{R})$ is also Fréchet-differentiable, then F is called twice Fréchet-differentiable. We denote by $\mathrm{d}F^{(2)}(u;\cdot,\cdot) \in \mathscr{L}(U,\mathscr{L}(U,\mathbb{R}))$ the second Fréchet derivative of F at u. The nth Fréchet derivative of F for any $n \in \mathbb{N}$ can be defined accordingly.

The following result generalises Taylor's theorem to functionals on Hilbert spaces. Analogous statements hold for operators between Banach spaces. For a detailed proof, the interested reader is referred to [14, Theorem 5].

If F is n-times Fréchet-differentiable, we have

$$\begin{aligned} F(u+h) = F(u) + \mathrm{d}F(u;h) + \frac{1}{2}\mathrm{d}^{(2)}F(u;h,h) + \dots \\ + \frac{1}{n!}\mathrm{d}F^{(n)}(u;\underbrace{h,\dots,h}_{n \text{ times}}) + \mathscr{O}(\|h\|_H^{n+1}) \quad \forall u,h \in U. \end{aligned} \tag{1.104}$$

1.2.3 Partial derivatives and the chain rule

In the derivation of the adjoint to a PDE, it will also be necessary to differentiate between partial and total derivatives. Define $F : U \times U \to \mathbb{R}$ with U an open subset of some Hilbert space H, and $u : U \to U$. Then

$$\partial F_m(u(m),m;h) = \lim_{\varepsilon \to 0^+} \frac{F(u(m),m+\varepsilon h) - F(u(m),m)}{\varepsilon}, \tag{1.105}$$

is the partial derivative of F with respect to m while

$$\mathrm{d}F_m(u(m),m;h) = \lim_{\varepsilon \to 0^+} \frac{F(u(m+\varepsilon h),m+\varepsilon h) - F(u(m),m)}{\varepsilon}, \tag{1.106}$$

is the corresponding total derivative. As is the case in Euclidean space, the partial and total derivatives are related by the chain rule, which in this case is given by

$$\mathrm{d}F_m(u(m),m;\cdot) = \partial F_u(u(m),m;\mathrm{d}u_m(m;\cdot)) + \partial F_m(u(m),m;\cdot). \tag{1.107}$$

Omitting function arguments, this produces the familiar form

$$\mathrm{d}F_m = \partial F_u \mathrm{d}u_m + \partial F_m. \tag{1.108}$$

1.3 The relation between primal and dual finite element spaces

As previously seen for Burgers' equation, a finite element problem can be formulated in terms of its residual. The root of the residual corresponds to a solution of the problem, which leads to the archetypal formulation of a finite element problem, find $u \in U$ such that

$$F(u;v) = 0 \qquad \forall v \in V, \tag{1.109}$$

where U and V are suitable finite element spaces. In general, we call U the trial function space and V the test function space.

The weak formulation is derived by multiplying the original PDE with a test function v and integrating over the domain. Due to the linearity of integration $F(u;v)$ is therefore linear in v for any given $u \in U$. Further, for many PDEs it is straightforward to show that $F(u;\cdot)$ is continuous and therefore that $F(u;\cdot) \in V^*$. If we recall the definition of the operator (or dual) norm, then (1.109) can be written: find $u \in U$ such that

$$\|F(u;\cdot)\|_* = 0. \tag{1.110}$$

Similarly, the archetypal linear problem, find $u \in U$ such that

$$a(u,v) = L(v) \qquad \forall v \in V, \tag{1.111}$$

can equivalently be written, find $u \in U$ such that

$$\|a(u,\cdot) - L(\cdot)\|_* = 0. \tag{1.112}$$

In other words the weak form of a PDE is actually an equation between linear operators, with equality defined in terms of the appropriate operator norm.

Naturally, this is also reflected in the nature of the assembled matrices and vectors which result from actually invoking the finite element method. Recall that $v \in V$ can be represented with respect to a basis $\mathscr{P} = \{\phi_1, \ldots, \phi_d\}$:

$$v = \sum_{i=1}^{d} v_i \phi_i. \tag{1.113}$$

When we represent a function within a finite element space as a vector, it is of course this set of coefficients v_i that we are storing within an implementation in code.

$\mathscr{P}$ also induces a unique dual basis for V^*, $\mathscr{P}^* = \{\phi_1^*, \ldots, \phi_d^*\}$. The defining property of the dual basis is its orthogonality to $\mathscr{P}$:

$$\phi_i^*(\phi_j) = \delta_{ij} \quad \forall i,j = 1,\ldots,d. \tag{1.114}$$

This immediately results in the fundamental identity relating the dual basis, $\mathscr{P}^*$, and the coefficients of the primal basis, $\mathscr{P}$:

$$\forall v \in V,\ v = \sum_{i=1}^{d} v_i \phi_i \iff v_i = \phi_i^*(v) \quad \forall i = 1,\ldots,d. \tag{1.115}$$

Conversely, assembling a finite element operator as a vector simply amounts to expressing the operator as a linear combination of the basis functions for the dual space. In this case, the primal basis functions define the coefficients:

$$\forall F \in V^*,\ F = \sum_{i=1}^{d} F_i \phi_i^* \iff F_i = F(\phi_i) \quad \forall i = 1,\ldots,d. \tag{1.116}$$

1.3.1 Example: application of assembled finite element operators

A straightforward consequence of the above is that we can explain why the application of a finite element operator to a function is equivalent to the (ℓ^2) dot product of the associated assembled vectors. Suppose we have $v(=\sum_i v_i\phi_i) \in V$ and $F(=\sum_i F_i\phi_i^*) \in V^*$. Then

$$\begin{aligned} F(v) &= \sum_{i=1}^{d} F_i\phi_i^* \left(\sum_{j=1}^{d} v_j\phi_j \right) \\ &= \sum_{i,j=1}^{d} F_i v_j \phi_i^*(\phi_j) \\ &= \sum_{i,j=1}^{d} F_i v_j \delta_{ij} \\ &= \sum_{i=1}^{d} F_i v_i \\ &= \mathbf{F}\cdot\mathbf{v}, \end{aligned} \tag{1.117}$$

where $\mathbf{F}$ and $\mathbf{v}$ are the vectors of basis function coefficients corresponding to F and v, respectively. It is helpful in this context to follow [21] and introduce an operator, $\mathscr{I} : \mathbb{R}^{|V|} \to V$, defined by

$$\mathscr{I}(\mathbf{v}) = \sum_{i=1}^{d} v_i\phi_i. \tag{1.118}$$

In other words, $\mathscr{I}^{-1}$ takes functions $v \in V$ to their representation as a vector of basis function coefficients. We define a similar operator for the dual space, $\mathscr{I}^* : \mathbb{R}^{|V|} \to V^*$:

$$\mathscr{I}^*(\mathbf{F}) = \sum_{i=1}^{d} F_i\phi_i^*. \tag{1.119}$$

1.3.2 The primal and dual norms for L^2 and H^1

We will return to the choice of an appropriate norm in a subsequent chapter, however it may be helpful in understanding the relationship between the primal function space V and its dual V^* to work through the evaluation of a particular operator norm. For this purpose, we choose the most basic norm on finite element spaces, L^2, and let trial and function space coincide. Recall that for $u \in V \subset L^2$,

$$\|u\|_{L^2}^2 = \int_\Omega u^2 \,\mathrm{d}x. \tag{1.120}$$

Equivalently, the L^2 inner product is given by:

$$\langle u, v\rangle_{L^2} = \int_\Omega uv \,\mathrm{d}x \quad \forall u, v \in V. \tag{1.121}$$

Simply by writing $u = \sum_i u_i \phi_i$, $v = \sum_j v_j \phi_j$, we can transform this into the matrix expression

$$\langle u, v\rangle_{L^2} = \mathbf{u}^{\mathrm{T}} \mathrm{M} \mathbf{v}, \tag{1.122}$$

where

$$\mathrm{M}_{ij} = \int_\Omega \phi_i \phi_j \,\mathrm{d}x, \tag{1.123}$$

is commonly referred to as the mass matrix. Now, for any $u \in V$, we can define a functional $F_u \in \mathscr{L}(L^2, \mathbb{R})$ via

$$F_u(\cdot) = \langle u, \cdot\rangle_{L^2}. \tag{1.124}$$

It immediately follows that

$$\mathscr{I}^{*-1}(F_u) = \mathbf{F}_u = \mathrm{M}\mathbf{u}. \tag{1.125}$$

Conversely, for any $F \in V^*$, the uniqueness of the Riesz representation guarantees that

$$\mathscr{I}^{-1}\left(\mathscr{R}_{L^2}(F)\right) = \mathrm{M}^{-1}\mathscr{I}^{*-1}(F) = \mathrm{M}^{-1}\mathbf{F}. \tag{1.126}$$

In other words, the L^2 Riesz representer of a functional F is obtained by multiplying the corresponding coefficient vector by the inverse mass matrix.

The Riesz representation theorem also enables us to understand the role of the mass matrix in evaluating the L^2 operator norm. Recall that, for any $F \in V^*$

$$\|F\|_{L^{2*}} = \|\mathscr{R}_{L^2}(F)\|_{L^2}, \tag{1.127}$$

which implies that

$$\begin{aligned} \|F\|^2_{L^{2*}} &= \langle \mathscr{I}(\mathrm{M}^{-1}\mathbf{F}), \mathscr{I}(\mathrm{M}^{-1}\mathbf{F})\rangle_{L^2} \\ &= \mathbf{F}^{\mathrm{T}}\mathrm{M}^{-1}\mathrm{M}\mathrm{M}^{-1}\mathbf{F} \\ &= \mathbf{F}^{\mathrm{T}}\mathrm{M}^{-1}\mathbf{F} \\ &= F\left(\mathscr{R}_{L^2}(F)\right). \end{aligned} \tag{1.128}$$

An analogous argument to that given above applies to finite element subspaces of H^1 with inner product given by

$$\langle u, v\rangle_{H^1} = \int_\Omega \nabla u \cdot \nabla v + uv \,\mathrm{d}x. \tag{1.129}$$

In this case, if $V \subset H^1$ and $\mathbf{F} = \mathscr{I}^{*-1}(F)$ with $F \in V^*$ then

$$\mathscr{R}_{H^1}(F) = \mathrm{A}^{-1}\mathbf{F}, \tag{1.130}$$

where

$$\mathrm{A}_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j + \phi_i\phi_j \,\mathrm{d}x. \tag{1.131}$$

1.3.3 Convergence criteria for Newton's method

Having established the relationship between finite element solutions and residuals, and their corresponding norms, we can now return to the question of convergence criteria for nonlinear solvers, which we postponed from Sect. 1.1.2.6.

One possibility is use the magnitude of the residual as the stopping criterion, possibly normalised by the magnitude of the residual at the initial guess. The residual measures the extent to which the current iteration fails to be a solution to the discretised PDE, and this approach essentially coincides with the usual approach for determining convergence in linear systems. Since the residual is an functional in the dual space to the solution, the natural norm in which to evaluate this residual is the dual norm appropriate to the finite element space under consideration. However, as shown in the previous section, this amounts to solving a matrix system, which seems an expensive operation given that the result merely indicates convergence and does not actually improve the solution in any way.

One widely adopted work-around for this solution is to employ a more conveniently calculated norm. The obvious and usual choice is to observe that the vector of coefficients of the residual vector is isomorphic to a vector in $\mathbb{R}^d$ and to employ the ℓ^2 norm of that vector:

$$\|F\|_{\ell^2} = \sqrt{\mathbf{F}\cdot\mathbf{F}}, \tag{1.132}$$

where $\mathbf{F} = \mathscr{I}^{*-1}(F)$. It is simple to show that this is indeed a norm on V^*. However, what has been lost by discarding the factor of M^{-1} or A^{-1} is the notion that the norm in which we measure the residual is a discrete approximation to the norm on the original, continuous, Sobolev space. In fact, as the example of the stiffness matrix in Sect. 1.1.1.7 shows, the discretisation is explicit in the mesh cell Jacobian entries that appear in the transformation of the integral measure and, in the H^1 case, in the gradient operator. This situation is, in fact, a close analogy of the problem that we analyse in much more depth in Chapt. 2. Rather than restate that analysis here, we will simply note that the impact of discarding the mesh operator from the norm is that the value of the ℓ^2 norm of the residual is entirely dependent on the mesh chosen. This has a number of undesirable consequences, including that the number of iterations required to achieve convergence is mesh-dependent, and that candidate solutions on different meshes which differ by arbitrarily small amounts in the primal norm, can result in residuals whose ℓ^2 norms differ by arbitrarily large factors. In short, the ℓ^2 norm is simply not a good indicator of the magnitude of the residual. Note that this problem arises through the infinite-dimensional formulation of the underlying problem. On finite-dimensional spaces, all norms are equivalent.

If the operator norm is inconvenient to evaluate, and other norms applied to the residual are poor indicators of convergence, how then should convergence be mea-

sured? One solution is to use the magnitude of the update step $\hat{u}$, which is an estimator for the error in the solution at the previous step of the iteration. Note that $\hat{u} \in V$ so the natural norm only involves multiplication by a sparse and easily assembled matrix. Given a tolerance $\varepsilon > 0$, one terminates the algorithm if

$$\frac{\|u_h^{k+1} - u_h^k\|_V}{\|u_h^k\|_V} < \varepsilon. \tag{1.133}$$

The best norm to use depends on the Sobolev space in which the solution is sought. For the examples we have considered, this is the H^1 norm. For a much more in-depth analysis of nonlinear solvers for numerical PDE problems, the reader is referred to [7]. The Riesz map also emerges as an important consideration in the preconditioning of the linear systems which emerge from the numerical solution of PDEs. The interested reader is referred to [21].

1.4 Adjoint and tangent linear equations

Recall the generic formulation of a PDE-constrained optimisation problem

$$\min_{\substack{u \in V \\ m \in M}} J(u,m) \tag{1.134}$$

$$\text{subject to } F(u,m;v) = 0 \quad \forall v \in V', \tag{1.135}$$

where $J : V \otimes M \to \mathbb{R}$ is the objective functional, $m \in M$ is the control variable, and $u \in V$ is the weak solution of the parametrised PDE. In the context of the adjoint equations (1.135) is referred to as the *forward model*. We assume that the forward model yields a unique solution for any control value, so that we can define the solution operator $u(\cdot) : M \to V$ which maps a control value to the associated solution of the forward model. Of course the explicit form of $u(m)$ is not known for many PDEs, but a finite element solver might be used instead as an approximation. Substituting the solution operator into (1.134) yields the *reduced functional*

$$\tilde{J}(m) = J\left(u(m), m\right), \tag{1.136}$$

and the associated *reduced optimisation problem*

$$\min_{m \in M} \tilde{J}(m). \tag{1.137}$$

Most efficient optimisation algorithms for solving (1.137) require evaluations of the functional gradient with respect to the control variable, i.e. $\mathrm{d}\tilde{J}_m(m;\cdot) = \mathrm{d}J_m(u(m),m;\cdot) \in M^*$. Table 1.1 lists common techniques for computing the gradient, together with an estimate of their computational expense. For PDE-constrained optimisation, the number of input parameters (the dimension of M) is typically large, typically ranging from 10^2 when optimising the positioning of wind turbines to

Method	Number of PDE solves
Finite difference approximation	i
Complex step approach	i
Tangent linear approach	i
Adjoint approach	o

Table 1.1: Number of PDE solves required to compute the derivative of a computer model with i input and o output parameters for different methods. The PDE solves are typically the principle computational cost factor, and hence provide a good estimate for the total expense. PDE-constrained optimisation problems have $o = 1$ and $i \gg 1$, and hence the adjoint approach is by far the most efficient method

$> 10^9$ when optimising the initial state of the ocean in the ECCO2 ocean model [34]. At the same time, the number of output values is one (the functional value). Examining table 1.1 reveals that the adjoint approach is the most effective option for evaluating the functional gradient as the required number of PDE solves is independent of the number of input parameters.

1.4.1 A finite-dimensional example

To understand how the adjoint approach computes the functional derivative so efficiently, let us consider a finite-dimensional example. Consider the functional

$$J(u,m) := j^T u + \frac{1}{2} m^T m, \tag{1.138}$$

with $j \in \mathbb{R}^n$ given and a parameter vector $m \in \mathbb{R}^m$. Furthermore, let $u \in \mathbb{R}^n$ be the solution of the linear system

$$Au = Bm, \tag{1.139}$$

with $A \in \mathbb{R}^{n \times n}$ invertible and $B \in \mathbb{R}^{n \times m}$.

Our aim is to compute the total derivative of the functional (1.138) with respect to m. Since A is invertible, we can solve (1.139) for u and substitute it into (1.138) to obtain the reduced functional:

$$\tilde{J}(m) := j^T A^{-1} B m + \frac{1}{2} m^T m. \tag{1.140}$$

From that, we can compute the total derivative with respect to m:

$$d\tilde{J}(m) = j^T A^{-1} B + m^T. \tag{1.141}$$

Depending on the order in which we evaluate these terms, we obtain the tangent linear or the adjoint approach:

Tangent linear approach

1. Solve the tangent linear system $A\mu = B$ for the tangent linear solution $\mu \in \mathbb{R}^{n\times m}$.
2. Evaluate the gradient with $j^T\mu + m^T \in \mathbb{R}^m$.

Adjoint approach

1. Solve the linear system $A^T\lambda = j$ for the adjoint solution $\lambda \in \mathbb{R}^n$.
2. Evaluate the gradient with $\lambda^T B + m^T \in \mathbb{R}^m$.

The two different approaches are visualised in figure 1.6. The computational cost of the tangent linear approach is dominated by the solution of the tangent linear system $A\mu = B$. Figure 1.6 reveals that the right hand side of this linear system is a matrix with m columns, corresponding to the number of parameters. One option to solve this system is to invert the matrix A explicitly (for example with a LU decomposition) and to perform a matrix-matrix multiplication to compute μ. Alternatively, m iterative solves can be performed for each column vector in B. However, in both cases, the computational cost scales linearly with m.

In contrast, the right hand side of the adjoint equation $A^T\lambda = j$ consists of a single column vector. Hence, the computation of the adjoint solution λ requires only one linear solve, independent of the dimension of m. Since the dominant computational cost is the solution of the linear systems, the adjoint approach yields a computational cost that is practically independent of m.

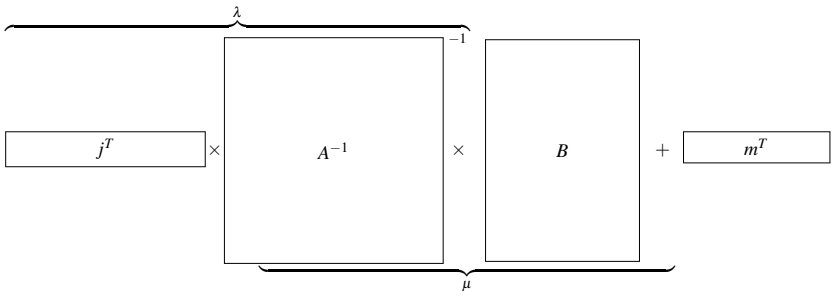


Fig. 1.6: The dimensions of the terms in the gradient (1.141) visualised. The tangent linear solution $\mu \in \mathbb{R}^{n\times m}$ requires the solution of m linear systems, while the adjoint solution $\lambda \in \mathbb{R}^n$ requires only one linear solve

1.4.2 The infinite-dimensional case

Let us return to the case where the function spaces in (1.135) are infinite-dimensional. For brevity we omit operator arguments, that is, we write $J \in \mathbb{R}$ instead of $J(u,m) \in \mathbb{R}$ and $F \in V'^*$ instead of $F(u,m;.) \in V'^*$.

Taking the derivative of Eq. (1.136) and applying the chain rule on its right hand side yields:

$$\mathrm{d}J_m = \partial J_u \mathrm{d}u_m + \partial J_m \in M^*. \tag{1.142}$$

Here we assumed that the solution operator $u(m)$ is continuously Fréchet differentiable, which follows if J and F are continuously Fréchet differentiable and if the linearised PDE operator $\partial F_u(u(m),m;.,.)$ is invertible for any $m \in M$, see [19, Sect. 1.4.2] and [19, Sect. 1.6]. If Eq. (1.142) was used directly to compute the functional gradient, all three terms on the right hand side would be needed. For the terms involving J this is straightforward: the functional of interest is typically given explicitly and its partial derivatives can therefore be readily derived. However, $\mathrm{d}u_m \in \mathscr{L}(M,V)$ is typically not known explicitly, since u is the solution of the PDE.

The state equation (1.135) provides us a way to compute $\mathrm{d}u_m$: Since $F(u(m),m) = 0$ for all $m \in M$, its derivative $d_m F(u(m),m)$ must be zero. Applying the chain rule on the left hand side yields:

$$\partial F_u \mathrm{d}u_m + \partial F_m = 0 \in M^*. \tag{1.143}$$

Since it was assumed that ∂F_u is invertible, we can substitute (1.143) into (1.142) to obtain an equation for the functional gradient:

$$\mathrm{d}J_m = -\overbrace{\partial J_u \underbrace{\partial F_u^{-1}}_{=\mu} \partial F_m}^{=\lambda^*} + \partial J_m. \tag{1.144}$$

The order in which the right hand side is evaluated leads to two different approaches: one involves solving the *tangent linear model* for μ, and the other one solves the *adjoint model* for $\lambda \in V'$.

1.4.2.1 The tangent linear approach

The tangent linear approach first solves the weak tangent linear model for $\mu \in \mathscr{L}(M,V)$:

$$\partial F_u(u,m;v,\mu) = \partial F_m(u,m;v) \quad \forall v \in V', \tag{1.145}$$

and then uses (1.144) to compute the functional derivative with

$$\mathrm{d}J_m(u,m;\cdot) = -\partial J_u(u,m;\mu(\cdot)) + \partial J_m(u,m;\cdot). \tag{1.146}$$

Note that the linearised forward model (1.143) and the tangent linear model (1.145) are equivalent apart from the sign, and hence $\mu = -\mathrm{d}u_m$.

1.4.2.2 The adjoint approach

The adjoint approach solves the equation: Find $\lambda \in V'$ such that:

$$\partial F_u^*(u,m;v,\lambda) = \partial J_u(u,m;v) \quad \forall v \in V. \tag{1.147}$$

or equivalently,

$$\partial F_u(u,m;\lambda,v) = \partial J_u(u,m;v) \quad \forall v \in V. \tag{1.148}$$

Here, the adjoint equation is stated in the familiar form of a variational problem, with a bilinear operator on the left hand side and a linear operator on the right hand side and can be solved with the finite element method. The adjoint operator is based on the linearisation of the original PDE operator, so its computational cost is expected to be equal or less than that of the original PDE.

Once the adjoint solution has been computed, the functional derivative is obtained with

$$\mathrm{d}J_m(u,m;\cdot) = -\partial F_m(u,m;\cdot,\lambda) + \partial J_m(u,m;\cdot). \tag{1.149}$$

1.4.2.3 Alternative approaches

There exist alternative approaches to compute the functional gradient. Two common methods are the finite difference and the complex-step approaches which will be briefly discussed in this section.

The finite difference approach approximates the functional gradient in the direction δm by computing the difference quotient. The central difference formula for example yields:

$$\mathrm{d}\tilde{J}_m(m;\delta m) = \frac{\tilde{J}(m+h\delta m) - \tilde{J}(m-h\delta m)}{2h} + \mathscr{O}(|h|^2) \quad \text{as } h \to 0.$$

The main advantage of the finite difference approach is its easy implementation: the model can be treated as a black box since only functional evaluations are required. However, the determination of a suitable step length h can be difficult: from a mathematical perspective, h should be chosen as small as possible to improve the approximation, but due to numerical cancellation the smallest suitable h value is bounded by round-off errors [27, pp. 166–169].

The complex-step approach [24] avoids this problem by using complex calculus. It considers the reduced functional in the complex plane and uses the Cauchy-Riemann equations to derive following approximation of the directional derivative:

$$\mathrm{d}\tilde{J}_m(m;\delta m) = \frac{\mathrm{Im}\left(\tilde{J}(m+ih\delta m)\right)}{h} + \mathscr{O}(|h|^2) \quad \text{as } h \to 0.$$

Since no difference operation is performed, this evaluation is not subject to subtractive cancellation errors. From an implementation perspective the complex-step approach is more intrusive, since the underlying code must be modified to support complex numbers.

Both the finite difference and the complex step approach only yield the derivatives in a particular direction. To obtain the full functional derivative, the directional derivatives for all basis functions in the parameter space must be computed separately. As a consequence the computational complexity increases linearly with the dimension of M. Nevertheless, the finite difference method is popular due to its straightforward implementation and is a useful verification tool.

1.4.3 Higher-order derivatives

It is possible to compute higher-order derivatives by recursively applying the adjoint or tangent linear approach. One application of second-order information is in the context of optimisation with PDE constraints, for example for design optimisation [16] and optimal control [18, 30]. For brevity we only state a short derivation of the second order adjoint equations here. A more detailed derivation of second-order adjoints can be found for example in [15, Sect. 13.4], [19, Sect. 1.6.5], [36, Sect. 4.6], and [5].

The action of the Hessian H is the transpose of the total derivative of the first derivative in a particular direction δm. Applying the total derivative to (1.149) and omitting the arguments for brevity, we find:

$$\begin{aligned}(H\delta m)^* = \mathrm{d}\left(\mathrm{d}J_m\right)_m \delta m = &-\dot{\lambda}^* \partial F_m \\ &- \lambda^* \mathrm{d}\left(\partial F_m\right)_m \delta m \\ &+ \mathrm{d}\left(\partial J_m\right)_m \delta m,\end{aligned} \tag{1.150}$$

where

$$\dot{\lambda} = \mathrm{d}\lambda_m \delta m \tag{1.151}$$

is the tangent linearisation of the adjoint solution λ in the direction δm. $\dot{\lambda}$ denotes the second order adjoint solution, which is derived by taking the total derivative of the adjoint equation (1.148):

$$\begin{aligned}\partial F_u^* \dot{\lambda} = &\left(\partial^2 J_{uu} \dot{u}\right)^* + \left(\partial^2 J_{mu} \delta m\right)^* \\ &- \left(\partial^2 F_{uu} \dot{u}\right)^* \lambda - \left(\partial^2 F_{mu} \delta m\right)^* \lambda.\end{aligned}$$

Evaluating the directional Hessian is computationally quite more expensive than a gradient evaluation. In particular, to compute the functional, only the forward PDE needs to be solved. To compute the functional's gradient, the forward and adjoint PDEs need to be solved. However, to compute the Hessian in one direction, the forward and adjoint PDEs, a tangent linear PDE and a second order adjoint PDE

need to be solved. This comparable high cost will play an important role for the design of optimisation algorithms in section 1.5.

1.4.4 Example: Adjoint Poisson's equation

Recall from (1.12) the variational form of the Poisson's equation with pure Neumann conditions: find $u \in V = H^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x - \int_{\partial\Omega} g_N v \,\mathrm{d}s = \int_\Omega f v \,\mathrm{d}x \quad \forall v \in V' = H^1(\Omega). \tag{1.152}$$

with $f \in L^2(\Omega)$ and together with the functional $\frac{1}{2}\int_\Omega u \cdot u \,\mathrm{d}x + \frac{\alpha}{2}\int_\Omega f^2 \,\mathrm{d}x$ with $\alpha \geq 0$.

We can write this problem in the canonical form (1.135) by defining following operators:

$$F(u,f;v) = \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x - \int_{\partial\Omega} g_N v \,\mathrm{d}s - \int_\Omega f v \,\mathrm{d}x, \tag{1.153}$$

$$J(u,f) = \frac{1}{2}\int_\Omega u \cdot u \,\mathrm{d}x + \frac{\alpha}{2}\int_\Omega f^2 \,\mathrm{d}x. \tag{1.154}$$

For the operators in the adjoint equation (1.148) we obtain

$$\partial F_u^*(u,m;v,\lambda) = \partial F_u(u,m;\lambda,v) = \int_\Omega \nabla v \cdot \nabla \lambda \,\mathrm{d}x, \tag{1.155}$$

$$\partial J_u(u,f;v) = \int_\Omega uv \,\mathrm{d}x. \tag{1.156}$$

Hence, the adjoint Poisson's equation for the functional is: Find $\lambda \in V'$ such that

$$\int_\Omega \nabla v \cdot \nabla \lambda \,\mathrm{d}x = \int_\Omega u \cdot v \,\mathrm{d}x \quad \forall v \in V, \tag{1.157}$$

or in strong form:

$$-\Delta\lambda = u \qquad \text{in } \Omega, \tag{1.158}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega. \tag{1.159}$$

Given λ, the functional derivative can be evaluated as:

$$\mathrm{d}J_f(u,f;\cdot) = -\partial F_f(u,f;\cdot,\lambda) + \partial J_f(u,f;\cdot) = \int_\Omega (\lambda + \alpha f)\cdot \,\mathrm{d}x. \tag{1.160}$$

Similarly, we could also evaluate the functional derivative with respect to the Neumann boundary values:

$$\mathrm{d}J_{g_N}\cdot = -\partial F_{g_N}\cdot + \partial J_{g_N}\cdot = \int_{\partial\Omega} \lambda \cdot \,\mathrm{d}s. \tag{1.161}$$

1.4.5 Example: Adjoint Burgers' equation

Next we derive the adjoint equation of the time-dependent viscous Burgers' equation with Neumann boundary conditions. The weak formulation of the Burgers' equation was derived in Sect. 1.1.2.1: find $u \in V = H^1(\Omega)^2$ such that

$$\int_\Omega \left(\frac{\partial u}{\partial t} \cdot v + (u\cdot\nabla)\, u \cdot v + \eta \nabla u : \nabla v \right) \mathrm{d}x = 0 \quad \forall v \in V' = H^1(\Omega)^2. \tag{1.162}$$

At this point we have two choices. Either we discretise the equation in time as in section 1.1.2.2, and derive the adjoint from the semi-discretised equation. Or derive the adjoint equation directly from the non-discretised form (1.162). We follow the latter strategy and extend the variational formulation such that our test and trial spaces are functions in time and space. Consequently, we extend the variational form by a time-integral and incorporate the initial condition into the variational formulation: Find $u \in L^2(0,T;V) \cap H^1(0,T;V^*)$ such that

$$\begin{aligned} &\int_0^T \int_\Omega \left(\frac{\partial u}{\partial t} \cdot v + (u\cdot\nabla)\, u \cdot v + \eta \nabla u : \nabla v \right) \mathrm{d}x\,\mathrm{d}t \\ &+ \int_\Omega (u(0) - u_0)\cdot v(0)\,\mathrm{d}x = 0 \quad \forall v \in L^2(0,T;V') \cap H^1(0,T;V'^*). \end{aligned} \tag{1.163}$$

The adjoint PDE operator is obtained by linearising in direction λ, and forming its transpose by swapping test and trial functions:

$$\int_0^T \int_\Omega \left(\frac{\partial v}{\partial t} \cdot \lambda + (v\cdot\nabla)\, u \cdot \lambda + (u\cdot\nabla)\, v \cdot \lambda + \eta \nabla v : \nabla \lambda \right) \mathrm{d}x\,\mathrm{d}t + \int_\Omega \lambda(0)\cdot v(0)\,\mathrm{d}x. \tag{1.164}$$

Integrating the time-derivative term by parts yields:

$$\int_0^T \int_\Omega \left(-\frac{\partial \lambda}{\partial t} \cdot v + (v\cdot\nabla)\, u \cdot \lambda + (u\cdot\nabla)\, v \cdot \lambda + \eta \nabla v : \nabla \lambda \right) \mathrm{d}x\,\mathrm{d}t + \int_\Omega \lambda(T)\cdot v(T)\,\mathrm{d}x \tag{1.165}$$

Note that the integration by parts results in the term $-\int_\Omega \lambda(0)\cdot v(0)\,\mathrm{d}x$ which cancels with the last term in (1.164). Eq. (1.165) is the weak adjoint operator for Burgers' equation and can be discretised and solved with the finite element method. The strong form can also be obtained through integration by parts. For a functional $J(u,m)$ one obtains:

$$
\begin{aligned}
-\frac{\partial \lambda}{\partial t} - (u\cdot\nabla)\lambda + (\nabla u)^*\lambda - \eta\Delta\lambda &= \partial J_u^* \quad \text{in } [0,T]\times\Omega,\\
(n\cdot\nabla)u &= 0 \quad \text{on } [0,T]\times\partial\Omega,\\
\lambda &= 0 \quad \text{on } \{T\}\times\Omega.
\end{aligned}
\tag{1.166}
$$

The adjoint of Burgers' equation (1.166) reveals some general properties about the adjoint of a time-dependent problem: the resulting PDE is linear with initial conditions at the end of the time interval. Consequently it is solved backwards in time. Furthermore, the adjoint PDE depends on the forward solution u, and therefore the forward model must be solved beforehand.

The derivation and implementation of the adjoint system can be laborious for complex PDEs. To avoid the need for explicit adjoint derivations, special software tools have been developed that aim automate the derivation process.

1.4.6 Taylor tests for adjoint implementations

Verification, or correctness testing, is an essential component of the implementation of any numerical algorithm on a computer. This is particularly true for the technically complex software required to implement the numerical solver for an adjoint PDE. A Taylor test computes approximate convergence rates of Taylor remainders which can be used to compare to theoretical convergence rates based on Taylor's theorem. Solving the adjoint PDE is a mechanism for evaluating the Fréchet derivative of the reduced functional with respect to the control parameters, so the applicable form of Taylor's theorem is that given in Sect. 1.2.

Recall that $\tilde{J}(m) := J(u(m),m)$ is the reduced functional and let δm be an arbitrary perturbation of the control m. Applying Taylor's theorem to this situation, we find for $\mathbb{R} \ni h \to 0$

$$
\left|\tilde{J}(m+h\delta m) - \tilde{J}(m)\right| = \mathcal{O}(h), \text{ and} \tag{1.167}
$$

$$
\left|\tilde{J}(m+h\delta m) - \tilde{J}(m) - h\,\mathrm{d}\tilde{J}(m;\delta m)\right| = \mathcal{O}(h^2). \tag{1.168}
$$

The expression in (1.167) is referred to as the first-order Taylor remainder while that in (1.168) is called the second-order Taylor remainder.

Let us consider the situation where computer implementations of $\tilde{J}$ and $\mathrm{d}\tilde{J}$ are given and it is claimed that $\mathrm{d}\tilde{J}$ correctly implements the Fréchet derivative of $\tilde{J}$. One may numerically test this claim by computing the second-order Taylor remainder for a fixed but arbitrary choice of δm, and an exponentially decreasing sequence of values of h. If the implementation is correct then for sufficiently small h, the Talyor remainder will decrease as $\mathcal{O}(h^2)$.

Similarly, one can employ higher order Taylor remainders to test implementations of higher order Fréchet derivatives. For example the third-order remainder can be used to test implementations of the second derivative:

$$\left|\tilde{J}(m+h\delta m)-\tilde{J}(m)-h\,\mathrm{d}\tilde{J}(m;\delta m)-\frac{1}{2}h^2\,\mathrm{d}^{(2)}\tilde{J}(m;\delta m,\delta m)\right|=\mathscr{O}(h^3). \quad (1.169)$$

Clearly in this case the expected convergence rate is $\mathscr{O}(h^3)$. Since the Taylor test for the second derivative requires a correctly implemented first derivative, it is highly advisable to first conduct a Taylor test of the first derivative.

1.5 Optimisation methods

The preceding sections introduced the theory and practice of solving PDE problems using the finite element method, and the efficient computation of derivatives of functionals of the associated PDE solutions. On this basis it is now possible to introduce algorithms which employ these solution techniques and the resulting gradient information to solve PDE-constrained optimisation problems. Here we will introduce several iterative approaches which are well-known in the optimisation community. In contrast with most introductory works, we will formulate these algorithms in a general Hilbert space, rather than assuming that the optimisation problem is set in Euclidean space.

1.5.1 Steepest descent method

The steepest descent method is an optimisation algorithm which is based upon the observation that a differentiable function decreases fastest in the proximity of the current iterate in the direction of its negative gradient. This principle naturally generalises to functionals. As before, we formulate the optimisation problem in terms of the reduced functional:

$$\min_{m\in M}\tilde{J}(m) \quad (1.170)$$

where $\tilde{J}:M\to\mathbb{R}$ is the reduced functional defined over a control parameter drawn from a Hilbert space M. For any sufficiently small $\alpha>0$ and with $m':=m-\alpha\mathscr{R}(\mathrm{d}\tilde{J}(m;\cdot))$ for any $m\in M$ not a local minimiser of $\tilde{J}$, we obtain $\tilde{J}(m')<\tilde{J}(m)$. Recall that $\mathscr{R}(\mathrm{d}\tilde{J}(m;\cdot))\in M$ is the Riesz representer of the the Fréchet derivative of $\tilde{J}$ at point m. Iterating this approach yields

$$m_{k+1}=m_k-\alpha_k\mathscr{R}(\mathrm{d}\tilde{J}(m_k;\cdot))\quad\forall k\in\mathbb{N}_0, \quad (1.171)$$

such that $\tilde{J}(m_{k+1})\leq\tilde{J}(m_k)$.

If $\mathrm{d}\tilde{J}:M\to M^*$ is uniformly continuous on bounded sets and $\tilde{J}$ is uniformly convex, there is a unique global minimum. If further α_k satisfies the so called Wolfe conditions, the sequence $\{m_k\}$ converges strongly to the global minimum [12].

In Eq. (1.171), there are several possible choices for the step size length α_k. The simplest is $\alpha_k = \alpha \ \forall k \in \mathbb{N}_0$ for a sufficiently small, fixed $\alpha > 0$. Another choice, which is introduced next, is to apply a so called exact line search.

The Taylor series of $\tilde{J}(m_{k+1})$ around the point m_k is given by

$$\begin{aligned}\tilde{J}(m_{k+1}) &= \tilde{J}\left(m_k + \alpha_k \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)\right) && (1.172)\\ &= \tilde{J}(m_k) - \alpha_k \mathrm{d}\tilde{J}(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)) \\ &\quad + \frac{\alpha_k^2}{2}\mathrm{d}\tilde{J}^{(2)}(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right), \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)) && (1.173)\\ &\quad + \alpha_k^3 \mathscr{O}(\| \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right) \|_M^3).\end{aligned}$$

Note that the last term on the right hand side vanishes in the case where f is quadratic, i.e. $\tilde{J}^{(n)} \equiv 0$ for any $n > 2$. Equivalently, simply dropping the error term results in a quadratic approximation to $\tilde{J}(m_{k+1})$. Differentiating this approximation with respect to α_k and setting the result equal to zero leads to

$$\hat{\alpha}_k = \frac{\mathrm{d}\tilde{J}\left(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)\right)}{\mathrm{d}\tilde{J}^{(2)}\left(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right), \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)\right)}. \tag{1.174}$$

Under suitable hypotheses, among others convexity, $\hat{\alpha}_k$ defines the unique step length which minimises the quadratic approximation of $\tilde{J}(m_{k+1})$. If $M = \mathbb{R}^n$, the positive definiteness of the Hessian of $\tilde{J}$ is sufficient. The iterative approach induced by the step length $\alpha_k = \hat{\alpha}_k$ is called exact line search. Algorithm 1 shows the pseudocode for the steepest descent method with an exact line search.

Algorithm 1: Steepest descent with exact line search in a Hilbert space M

Input: Initial m_0 , convergence tolerance $\varepsilon > 0$;
while $\|\mathscr{R}(\mathrm{d}\tilde{J}(m_k;\cdot))\|_M > \varepsilon$ **do**
  $\alpha_k \leftarrow \mathrm{d}\tilde{J}(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right)) / \mathrm{d}\tilde{J}^{(2)}(m_k; \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right), \mathscr{R}\left(\mathrm{d}\tilde{J}(m_k;\cdot)\right))$;
  $m_{k+1} \leftarrow m_k - \alpha_k \mathscr{R}(\mathrm{d}\tilde{J}(m_k;\cdot))$;
end

1.5.2 Inexact line search and the Wolfe conditions

The exact line search procedure requires the evaluation of the functional Hessian which, as is discussed below, may not be easily or affordably computable. Consequently, more heuristic approaches have been developed, along with appopriate criteria under which they generate convergent optimisation schemes.

All of the methods we will consider produce iterative updates to the control m, of the form:

$$m_{k+1} = m_k + \alpha_k p_k. \tag{1.175}$$

α_k, a scalar, is referred to as the *step length* while p_k, which lies in the same space as m, is the *search direction*. The Wolfe conditions [37] are conditions on α_k which, given suitable conditions on the functional to be minimised, are sufficient to show that the optimisation converges. A more complete discussion of the Wolfe conditions is to be found in [3].

The first condition is the sufficient decrease condition:

$$\tilde{J}(m_{k+1}) \leq \tilde{J}(m_k) + \lambda \alpha_k \mathrm{d}\tilde{J}(m_k; p_k) \tag{1.176}$$

for a small parameter $\lambda > 0$. This condition, also called the Armijo condition, effectively prevents large steps which only cause small functional decreases. The second condition is the curvature condition:

$$|\mathrm{d}\tilde{J}(m_{k+1}; p_k)| \leq \beta |\mathrm{d}\tilde{J}(m_k; p_k)| \tag{1.177}$$

A popular and effective inexact line search strategy is to progressively decrease candidate α_k values until these conditions are met. This approach is known as backtracking.

1.5.3 Line search Newton-CG

As the name suggests, the line search Newton conjugate gradient (Newton-CG) algorithm is Newton's method augmented by a line search for the update size, and employing the conjugate gradient method to invert the Hessian. Newton's method amounts to approximating the system to be optimised by the leading terms of the Taylor series and solving for the fixed point given by the Jacobian being zero:

$$\begin{aligned} 0 &= \mathrm{d}\tilde{J}(m_{k+1}; \cdot) && (1.178)\\ &= \mathrm{d}\tilde{J}(m_k + p_k; \cdot) && (1.179)\\ &\approx \mathrm{d}\tilde{J}(m_k + p_k; \cdot) + \mathrm{d}^{(2)}\tilde{J}(m_k; p_k, \cdot). && (1.180) \end{aligned}$$

The procedure is to solve (1.180) for the search direction p_k and then to employ a line search to establish the step size.

The Newton-CG algorithm in $\mathbb{R}^n$ is given in [26, Sect. 7.1], while its generalisation to functionals on Hilbert spaces, achieved by carefully applying the correct inner products and norms, is given in algorithm 2. In this algorithm, the inner iteration sequence converges to the Newton step p_k in (1.180) and at each outer iteration, a tolerance $\varepsilon_k > 0$ for the accuracy of the computed p_k is specified. In the inner iteration, which corresponds to the conjugate gradient method, the search directions are denoted by d_j. The fallback case for negative curvature, $\mathrm{d}^{(2)}\tilde{J}(m_k; d_j, d_j) \leq 0$, ensures that the search direction p_k is always descending.

Algorithm 2: Line Search Newton-CG in Hilbert space

Input: Initial m_0 ;
for $k = 0, 1, \ldots$ **do**
  Define tolerance $\varepsilon_k > 0$;
  $z_0 \leftarrow 0; r_0 \leftarrow \mathscr{R}(\mathrm{d}\tilde{J}(m_k;\cdot)); d_0 \leftarrow -r_0;$
  for $j = 0, 1, \ldots$ **do**
    if $\mathrm{d}^{(2)}\tilde{J}(m_k;d_j,d_j) \leq 0$ **then**
      if $j = 0$ **then**
        $p_k \leftarrow -\mathscr{R}(\mathrm{d}\tilde{J}(m_k;\cdot));$
        break;
      else
        $p_k \leftarrow z_j;$
        break;
      end
    end
    $\beta_j \leftarrow \langle r_j, r_j\rangle_M / \mathrm{d}^{(2)}\tilde{J}(m_k;d_j,d_j);$
    $z_{j+1} \leftarrow z_j + \beta_j d_j;$
    $r_{j+1} \leftarrow r_j + \beta_j \mathscr{R}(\mathrm{d}^{(2)}\tilde{J}(m_k;d_j,\cdot));$
    if $\|r_{j+1}\|_M < \varepsilon_k$ **then**
      $p_k \leftarrow z_{j+1};$
      break;
    end
    $\gamma_{j+1} \leftarrow \langle r_{j+1}, r_{j+1}\rangle_M / \langle r_j, r_j\rangle_M;$
    $d_{j+1} \leftarrow -r_{j+1} + \gamma_{j+1} d_j;$
  end
  Compute α_k using an appropriate line search;
  $m_{k+1} \leftarrow m_k + \alpha_k p_k;$
end

1.5.4 BFGS

Newton-CG and other Krylov-based methods require many evaluations of the exact Hessian of the objective functional. Recall that, for a PDE-constrained optimisation problem, this in turn requires composed solutions of a (potentially time-varying) adjoint problem. To address this issue, it may be possible to employ optimisation algorithms which avoid the need to evaluate the Hessian.

In its original, Euclidean formulation the Broyden-Fletcher-Goldfarb-Shanno method is an optimisation algorithm for problems of the type

$$\min_{x\in\mathbb{R}} f(x) \tag{1.181}$$

with $f : \mathbb{R}^n \to \mathbb{R}$.

The BFGS method is one of the most successful and applied algorithms in unconstrained nonlinear optimisation [26]. As a quasi-Newton method, it does not require evaluations of the Hessian of the objective function, but only of the gradient and of the objective function itself. An approximation for the Hessian is computed using

the difference between successive iterates and the corresponding gradient vectors. In this respect, quasi-Newton methods can be understood as a generalisation of the secant method for finding roots of first derivatives. One advantage of the Hessian approximation is that it is always positive definite while the Hessian itself might merely be positive semi-definite and therefore not invertible.

In a manner analogous with Newton's method, we consider a quadratic model for the objective function. At a current iterate $x_k \in \mathbb{R}^n$, this model is given by:

$$q_k(x) = f(x_k) + \nabla f(x_k) \cdot (x - x_k) + \frac{1}{2}(x - x_k)^T \cdot B_k \cdot (x - x_k), \tag{1.182}$$

where $B_k \in \mathbb{R}^{n \times n}$ denotes the approximation of the Hessian of f evaluated at m_k. Since q_k is convex, one easily computes the unique minimum in the substituted variable $p_k = x - x_k$ as

$$p_k = -B_k^{-1} \cdot \nabla f(x_k). \tag{1.183}$$

Taking (1.183) as the line search direction, one may proceed as in Newton's method to compute the next iterate x_{k+1} by

$$x_{k+1} = x_k + \alpha_k p_k,$$

where $\alpha_k > 0$ denotes the step length satisfying sufficient decrease and curvature conditions given by the Wolfe conditons [26, Sect. 3.1]. The major difference from Newton's method is that B_k is an approximation to the Hessian and not the Hessian itself. The update of the quadratic model with $k \to k+1$ is achieved by updating the Hessian approximation $B_k \to B_{k+1}$. In fact, the BFGS algorithm explicitly constructs and updates $H_k \equiv B_k^{-1}$. This avoids the need to solve the linear system in (1.183) at each iteration.

How, then, do we construct and update H_k? It is a direct consequence of Taylor's theorem in $\mathbb{R}^n$ that

$$\nabla^2 f(x_k) \cdot (x_{k+1} - x_k) \approx \nabla f(x_{k+1}) - \nabla f(x_k) \qquad \text{for } \|x_{k+1} - x_k\| \ll 1, \tag{1.184}$$

where $\nabla^2 f(x_k) \in \mathbb{R}^{n \times n}$ denotes the Hessian of f evaluated at x_k. We impose the analagous condition on B_{k+1} resulting in the so called secant equation,

$$B_{k+1} \cdot (x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k), \tag{1.185}$$

or, equivalently,

$$H_{k+1} \cdot (\nabla f(x_{k+1}) - \nabla f(x_k)) = x_{k+1} - x_k, \tag{1.186}$$

The Wolfe conditions on α_k are sufficient to guarantee a solution to (1.186) [26, Sect. 6.1].

Choosing H_{k+1} as the closest matrix H to H_k with respect to a weighted Frobenius norm under the constraints that H is symmetric and satisfies (1.186) yields the

unique solution:

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{1.187}$$

where

$$s_k = x_{k+1} - x_k = \alpha_k p_k, \tag{1.188}$$
$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k), \tag{1.189}$$
$$\rho_k = \frac{1}{y_k^T s_k}, \tag{1.190}$$
$$V_k = \mathbf{I} - \rho_k y_k s_k^T. \tag{1.191}$$

One could also define a BFGS algorithm based on B_k instead of H_k. Using the Sherman-Morrison-Woodbury formula [26, Appendix] applied to (1.187), we obtain for the Hessian approximation B_{k+1}:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}. \tag{1.192}$$

We refer to the Eqs. (1.187) and (1.192) as the BFGS updating formulae. Observe that if B_k is symmetric then the update formula guarantees that B_{k+1} will be too. For more details of the derivations above and for algorithm 3, we refer to [26, Chap. 6].

Algorithm 3: BFGS in $\mathbb{R}^n$

Input: Initial x_0, convergence tolerances $\varepsilon_\nabla, \varepsilon > 0$, initial inverse Hessian approximation H_0;
$k \leftarrow 0$
while $\|\nabla f_k\| > \varepsilon_\nabla$ *and* $(f_k - f_{k+1})/\max(|f_{k+1}|, |f_k|, 1) > \varepsilon$ **do**
  $p_k \leftarrow -H_k \nabla f_k$;
  Compute α_k using an appropriate line search;
  $s_k \leftarrow \alpha_k p_k$;
  $x_{k+1} \leftarrow x_k + s_k$;
  $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$;
  Compute H_{k+1} according to (1.187);
  $k \leftarrow k+1$
end

1.5.4.1 BFGS in Hilbert spaces

We now return the the problem of minimising the reduced functional:

$$\min_{m \in M} \tilde{J}(m) \tag{1.193}$$

where M is a Hilbert space with inner product $\langle\cdot,\cdot\rangle_M$ and $\tilde{J}$ is assumed to be a twice continuously Fréchet-differentiable real-valued functional. The extension of the BFGS method to this situation simply requires careful attention to the appropriate application of this inner product. The quadratic model now becomes:

$$q_k(p) = \tilde{J}_k + y_k(p) + \frac{1}{2}B_k(p,p), \tag{1.194}$$

where $B_k \in \mathscr{L}(M,M^*)$ is invertible and $y_k = \mathrm{d}\tilde{J}(m_k) \in M^*$. Note that we can consider functions in $\mathscr{L}(M,M^*)$ to have either one argument in M and return values in M^*, or to have two arguments in M and return values in $\mathbb{R}$. We will use whichever version is most applicable in a given circumstance. The linear and continuous operator B_k is an approximation of the second Fréchet derivative of $\tilde{J}$. In order to find a p_k such that

$$q_k(p_k) \leq q_k(p) \quad \forall p \in M, \tag{1.195}$$

it is necessary that

$$\mathrm{d}q_k(p_k;\cdot) = y_k(\cdot) + B_k(p,\cdot) = 0. \tag{1.196}$$

This is typically referred to as the first order optimality condition. Therefore, the search direction we require is

$$p_k = -B_k^{-1}(y_k), \tag{1.197}$$

where $B_k^{-1} \in \mathscr{L}(M^*,M)$. The linearity and boundedness of B_k^{-1} can be demonstrated as follows. For any $m,n \in M$, there are $m^*,n^* \in M^*$ such that $B_k^{-1}m^* = m$ and $B_k^{-1}n^* = n$. Using the linearity of B_k yields

$$B_k(m+\gamma n) = B_k m + \gamma B_k n \tag{1.198}$$

$$\Leftrightarrow \quad B_k(B_k^{-1}m^* + \gamma B_k^{-1}n^*) = m^* + \gamma n^* \quad \forall \gamma \in \mathbb{R}. \tag{1.199}$$

Applying B_k^{-1} from the left hand side shows that B_k^{-1} is linear. Further, the open mapping theorem states that any surjective bounded linear operator between two Banach spaces is an open map, i.e. maps open sets onto open sets. Thus, for any $U \subset M$ open we have $(B_k^{-1})^{-1}(U) = B_k(U)$ open in M^*. Hence, B_k^{-1} is continuous.

A symmetric rank two update that meets the secant condition is the Hilbert space version of the BFGS formula (1.192), given by

$$B_{k+1}(m,n) = B_k(m,n) - \frac{B_k(s_k,m)B_k(s_k,n)}{B_k(s_k,s_k)} + \frac{y_k(m)y_k(n)}{y_k(s_k)}, \tag{1.200}$$

where the function arguments are $m,n \in M$. The updating formula corresponding to the approximation H_{k+1} of the Hessian inverse in (1.187) is given by

$$H_{k+1}(m^*) = H_k(m^*) - \frac{m^*(s_k)}{y_k(s_k)} H_k(y_k) - \frac{y_k(H_k(m^*))}{y_k(s_k)} s_k \tag{1.201}$$

$$+ \frac{m^*(s_k) y_k(H_k(y_k))}{(y_k(s_k))^2} s_k + m^*(s_k) s_k \tag{1.202}$$

$$= \left(\mathrm{Id}_M - \frac{s_k}{y_k(s_k)} y_k \right) \left(H_k(m^*) - H_k(y_k) \frac{m^*(s_k)}{y_k(s_k)} \right) + m^*(s_k) s_k \tag{1.203}$$

$$= \left(\mathrm{Id}_M - \frac{s_k}{y_k s_k} y_k \right) \left(H_k \left(m^* - \frac{m^*(s_k)}{y_k(s_k)} y_k \right) \right) + m^*(s_k) s_k \tag{1.204}$$

$$= \mathscr{V}_k \left(H_k \left(\mathscr{V}_k^*(m^*) \right) \right) + m^*(s_k) s_k, \tag{1.205}$$

with $m^* \in M^*$ arbitrary and $H_k \in \mathscr{L}(M^*, M)$. Here, $\mathrm{Id}_M : M \to M$ is the identity operator on the space M. Further, $\mathscr{V}_k : \mathscr{L}(M, M)$ and $\mathscr{V}_k^* : \mathscr{L}(M^*, M^*)$ are given by

$$\mathscr{V}_k(m) = m - \frac{s_k}{y_k(s_k)} y_k(m), \tag{1.206}$$

$$\mathscr{V}_k^*(m^*) = m^* - \frac{m^*(s_k)}{y_k(s_k)} y_k, \tag{1.207}$$

with $m \in M$ and $m^* \in M^*$ arbitrary.

Using the Riesz representation theorem, formula (1.200) can be written as

$$B_{k+1}(m,n) = B_k(m,n) - \frac{\langle \mathscr{R}(B_k(s_k)), m \rangle_M \langle \mathscr{R}(B_k(s_k)), n \rangle_M}{\langle \mathscr{R}(B_k s_k), s_k \rangle_M} + \frac{\langle \mathscr{R}(y_k), m \rangle_M \langle \mathscr{R}(y_k), n \rangle_H}{\langle \mathscr{R}(y_k), s_k \rangle_M}. \tag{1.208}$$

Of course we are primarily interested in the case where H is finite-dimensional. In the particular case where $M = \mathbb{R}^n$, we have $\langle a, b \rangle_M = a^T b$ and $\mathscr{R}(a) = a^T$. In this special case (1.208) is equivalent to (1.192). In contrast, if M is a finite-dimensional subspace of, for example, H^1 or L^2 then (1.208) demands that the corresponding inner product is employed.

The question arises which algorithm among all Hilbert space BFGS versions, or equivalently, what inner product to choose for numerically solving an optimisation problem. Letting ourselves be led by the structure of the problem , it is natural to choose the inner product space that corresponds to the space of the optimisation controls. The same space should be the basis for any finite-dimensional numerical method.

It turns out that optimisation algorithms that do not respect the natural and correct inner product of the control space lead to mesh-dependent and generally suboptimal solutions. Chapt. 2 and Chap. 3 are dedicated to a deeper discussion of these phenomena.

1.5.4.2 Limited memory Hessian approximation

The inverse Hessian approximation, H_k, introduced in the previous section for $\mathbb{R}^n$ is a dense $n \times n$ matrix. The storage of this matrix is impractical for all but the smallest problems. Fortunately, this is also unnecessary since only the action of this matrix is required. Instead, the sequence of vectors y_k and s_k can be stored, and $-H_k \nabla f_k$ computed on the fly using the recursion formular for H_k without ever constructing the matrix. Of course this still requires an increasingly large number of vectors to be stored and the computation of the matrix action becomes increasingly expensive. In contrast, [3] present an approach in which only information from the most recent iterations is stored. The justification for this approach, which is similar in character to restarted GMRES, is that the vectors essentially encode curvature information, and that the curvature information from the most recent iterations is likely to have the greatest impact on the Hessian behaviour of the current iteration [26, Sect. 7.2].

By repeated application of the BFGS formula (1.187), one can easily see that H_k is given recursively by

$$\begin{aligned} H_k = &(V_{k-1}^T \cdot \ldots \cdot V_{k-m}^T) H_{k-m} (V_{k-m} \cdot \ldots \cdot V_{k-1}) \\ &+ \rho_{k-m} (V_{k-1}^T \cdot \ldots \cdot V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} \cdot \ldots \cdot V_{k-1}) \\ &+ \rho_{k-m+1} (V_{k-1}^T \cdot \ldots \cdot V_{k-m+2}^T) s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} \cdot \ldots \cdot V_{k-1}) \\ &\vdots \\ &+ \rho_{k-2} V_{k-1}^T s_{k-2} s_{k-2}^T V_{k-1} \\ &+ \rho_{k-1} s_{k-1} s_{k-1}^T, \end{aligned} \tag{1.209}$$

where $1 \leq m \leq k$ and H_{k-m} is the approximation of the Hessian inverse at the $(k-m)$th iterate. The main idea of the limited BFGS method is now to replace H_{k-m} in (1.209) by some initialisation matrix $\gamma_k \mathbf{I}$ with $\gamma_k > 0$. The reader is referred to [3, Sect. 3] and [4, Sect. 3], for a more compact representation of H_k under further curvature conditions.

In a Hilbert space setting, (1.209) becomes

$$
\begin{aligned}
H_k(m^*) = {} & (\mathscr{V}_{k-1} \circ \ldots \circ \mathscr{V}_{k-m}) \left(H_{k-m}(\mathscr{V}^*_{k-m} \circ \ldots \circ \mathscr{V}^*_{k-1}(m^*)) \right) \\
& + \frac{\mathscr{V}_{k-1} \circ \ldots \circ \mathscr{V}_{k-m+1}}{y_{k-m} s_{k-m}} \left(s_{k-m} \cdot \left(\mathscr{V}^*_{k-m+1} \circ \ldots \circ \mathscr{V}^*_{k-1}(m^*) \right) (s_{k-m}) \right) \\
& + \frac{\mathscr{V}_{k-1} \circ \ldots \circ \mathscr{V}_{k-m+2}}{y_{k-m+1} s_{k-m+1}} \left(s_{k-m+1} \cdot \left(\mathscr{V}^*_{k-m+2} \circ \ldots \circ \mathscr{V}^*_{k-1}(m^*) \right) (s_{k-m+1}) \right) \\
& \vdots \\
& + \frac{\mathscr{V}_{k-1}}{y_{k-2} s_{k-2}} \left(s_{k-2} \cdot \mathscr{V}^*_{k-1}(m^*)(s_{k-2}) \right) \\
& + \frac{s_{k-1} m^*(s_{k-1})}{y_{k-1} s_{k-2}},
\end{aligned}
\tag{1.210}
$$

with $m^* \in M^*$ arbitrary.

1.5.5 Primal log-barrier interior point method

The concept of barrier methods for constrained optimisation relies on reformulating the constrained problem as a sequence of unconstrained problems. In that, a composite function is minimised which accounts both for the original objective function as well as the constraints. The interior point method used in 3.5 is closely related to the classical barrier interior point method summarised in [38], which is introduced in what follows. Consider the following inequality constrained nonlinear optimisation problem

$$
\begin{aligned}
& \min_{m \in M} \tilde{J}(m) \\
& \text{subject to } c_i(m) \geq 0,
\end{aligned}
\tag{1.211}
$$

where $\tilde{J} : M \to \mathbb{R}$ and $c_i : M \to \mathbb{R}$ for $i = 1, \ldots, n$. The optimisation method introduced in the following aims on solving (1.211) iteratively by requiring their iterates to belong to the interior of the feasible region, i.e. to satisfy the inequality constraints strictly. For (1.211), we can define the so called logarithmic barrier functional $\beta^\mu : M \to \mathbb{R}$ by

$$
\beta^\mu(m) = \tilde{J}(m) - \mu \sum_{i=1}^{n} \log\left(c_i(m)\right), \tag{1.212}
$$

where $\mu > 0$ is referred to as the barrier parameter. Note that the logarithmic terms are well-defined for $c_i(m) > 0$ but tend to infinity for $c_i(m)$ approaching zero. Intuitively, for small μ the minimum of β^μ is close to a solution of (1.211). In the case $H = \mathbb{R}^n$ and under sufficient optimality conditions on a minimiser $\hat{m}$ of (1.211), one can show ([38]) that for a monotonically decreasing sequence of sufficiently

small values of μ, there is a sequence m_μ of unconstrained minimisers of the barrier function (1.212) with

$$\lim_{\mu \to 0} m_\mu = \hat{m}. \tag{1.213}$$

Under appropriate smoothness assumptions, the sequence $\{x_\mu\}$ of points define a smooth curve and converges non-tangentially from the strict interior of the feasible region to the minimiser $\hat{m}$ [38].

The Fréchet derivative of the logarithmic barrier functional with respect to $m \in H$ is given by

$$\mathrm{d}\beta^\mu(m;\cdot) = \mathrm{d}\tilde{J}(m;\cdot) - \sum_{i=1}^{n} \frac{\mu}{c_i(m)} \mathrm{d}c_i(m;\cdot). \tag{1.214}$$

Using the product rule for differentiation, the barrier Hessian is computed as

$$\mathrm{d}^{(2)}\beta^\mu(m;\cdot,\cdot) = \mathrm{d}^{(2)}\tilde{J}(m;\cdot,\cdot) - \sum_{i=1}^{n} \frac{\mu}{c_i(m)} \mathrm{d}^{(2)}c_i(m;\cdot,\cdot) \tag{1.215}$$

$$+ \sum_{i=1}^{n} \frac{\mu}{c_i^2(m)} \mathrm{d}c_i(m,\cdot) \cdot \mathrm{d}c_i(m,\cdot) \tag{1.216}$$

To compute the step $p_k \in H$ to move from a current iterate m_μ^k to m_μ^{k+1}, i.e.

$$m_\mu^{k+1} = m_\mu^k + p_k, \tag{1.217}$$

we apply Newton's method to the quadratic model of the barrier function. This leads to solving the so called primal Newton barrier equation

$$\mathrm{d}^{(2)}\beta^\mu(m;p_k,\cdot) = -\mathrm{d}\tilde{J}(m;\cdot) + \sum_{i=1}^{n} \frac{\mu}{c_i(m)} \mathrm{d}c_i(m;\cdot). \tag{1.218}$$

Iterating this leads to an approximation of a stationary point of the barrier function (1.212). In the case $H = \mathbb{R}^n$ and under suitable differentiability assumption on β^μ, it can be shown that $\{m_\mu^k\}$ indeed converges to a stationary point and thus satisfies the necessary conditions to minimise the barrier function.

1.6 Optimal control of the Poisson equation

In this section, the principle of PDE-constrained optimisation is illustrated with a simple and generic example. The problem chosen is the optimal control of a system constrained by the Poisson equation. Physically, this problem can be interpreted as finding the ideal heat source in order to translate the state of the system into a desired temperature profile. Mathematically, the problem is to minimise the following

tracking type functional

$$\min_{m\in M} \frac{1}{2}\|u-d\|^2_{L^2(\Omega)} + \frac{\alpha}{2}\|m\|^2_M, \tag{1.219}$$

subject to the Poisson equation with Dirichlet boundary conditions given by

$$-\Delta u = m \quad \text{in } \Omega \tag{1.220}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{1.221}$$

where $\Omega = [0,1]^2$, $M = L^2(\Omega)$ or $H^1(\Omega)$, $u : \Omega \to \mathbb{R}$ is the unknown temperature, $m \in M$ is the unknown control function acting as source term ($m(x) > 0$ corresponds to heating and $m(x) < 0$ corresponds to cooling), $d \in L^2(\Omega)$ is the given desired temperature profile and $\alpha \in [0,\infty)$ is a Tikhonov regularisation parameter. For a proof of existence and uniqueness of the solution, we refer to [29, Sect. 1.5]. Using the formulation from 1.4, the objective functional for this problem is

$$J(u,m) := \frac{1}{2}\|u-d\|^2_{L^2(\Omega)} + \frac{\alpha}{2}\|m\|^2_M. \tag{1.222}$$

An iterative approach towards a solution is achieved using methods introduced previously. In a first step, a solution u_0 to the PDE constraints for a given user-defined initial control m_0 is computed using finite elements. In a second step, the Fréchet derivative of the reduced functional $\tilde{J}$ of J with respect to m_0 is computed using the adjoint approach. Using the derivative of $\tilde{J}$, an optimisation method is then applied in a third step, producing a new, better control m_1. This procedure is then iterated until a convergence criterion is met.

Based on this approach, a numerical solution of (1.219) satisfying (1.220, 1.221) using linear Lagrange finite elements for both m and u can be computed. Results are shown in Fig. 1.7. The parameters for the problem are

$$\Omega = [0,1]^2, \tag{1.223}$$

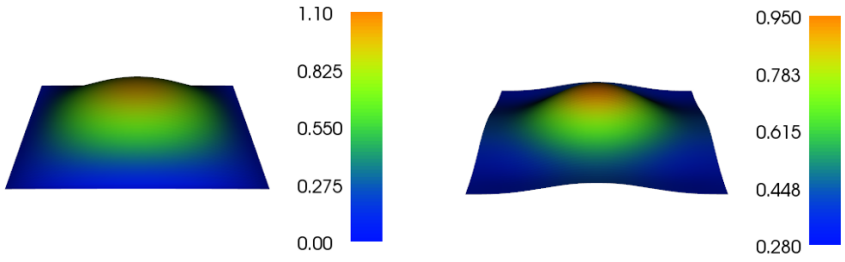
$$d(x_1,x_2) = \frac{1}{2\pi^2}\sin(\pi x_1)\sin(\pi x_2), \tag{1.224}$$

$$\alpha = 10^{-6}, \tag{1.225}$$

$$m_0 = 0. \tag{1.226}$$

The underlying optimisation method is an implementation of the limited memory BFGS algorithm from Sect. 1.5.4 provided by Moola [28]. While many widely used optimistation packages assume that the problem is posed in $\mathbb{R}^n$, this implementation respects the native structure of problems posed in arbitrary Hilbert spaces. In other words, the inner product native to the control space, is applied in the determination of search directions and convergence criteria. Respecting the inner products leads to mesh-independent convergence of the optimisation algorithm, which is explored in more detail in Chap. 2. For solving the finite element problem, the FEniCS frame-

work [23] was used. Similar to Firedrake, FEniCS enables the automated solution of differential equations. For an open access FEniCS tutorial we refer to [22]. Further, we made use of dolfin-adjoint [10] for the computation of adjoints. For the Python code used to produce the numerical solutions in this section, we refer to [33].



(a) $M = L^2$ (b) $M = H^1$

Fig. 1.7: Plots of $\hat{m}$ minimising (1.219) under the PDE-constraints (1.220, 1.221)

1.7 References

[1] Hans Wilhelm Alt. *Linear functional analysis: an application-oriented introduction*. Springer, 2016.

[2] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. 3rd Edition, Springer Science+Business Media, LLC, 2008.

[3] J. Nocedal Byrd R. H. and R. B. Schnabel. "Representations of quasi-Newton Matrices and their Use in Limited Memory Methods". In: *Mathematical Programming* 63 (1994), pp. 129–156.

[4] Richard H Byrd et al. "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[5] Alexandru Cioaca, Mihai Alexe, and Adrian Sandu. "Second-order adjoints for solving PDE-constrained optimization problems". In: *Optimization Methods and Software* 27.4-5 (2012), pp. 625–653. DOI: 10.1080/10556788.2011.610455.

[6] Timothy A Davis. *Direct methods for sparse linear systems*. Vol. 2. Siam, 2006.

[7] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Vol. 35. Springer Series in Computational Mathematics. Springer, 2011.

[8] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. USA: Oxford University Press, 2005.

[9] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press (UK), 2014.

[10] P. E. Farrell et al. "Automated derivation of the adjoint of high-level transient finite element programs". In: *SIAM Journal on Scientific Computing* (2012). accepted.

[11] *Firedrake documentation*. URL: `http://firedrakeproject.org/documentation.html`.

[12] Fernando Andrés Gallego, John Jairo Quintero, and Juan Carlos Riano. "Convergence of the steepest descent method with line searches and uniformly convex objective in reflexive Banach spaces". In: *Mathematical Communications* 20.2 (2015), pp. 161–173.

[13] Michael Ghil and Paola Malanotte-Rizzoli. "Data Assimilation in Meteorology and Oceanography". In: *Advances in Geophysics*. Ed. by Renata Dmowska and Barry Saltzman. Vol. 33. Elsevier, 1991, pp. 141–266. DOI: `10.1016/S0065-2687(08)60442-2`.

[14] Lawrence M Graves. "Riemann integration and Taylor's theorem in general analysis". In: *Transactions of the American Mathematical Society* 29.1 (1927), pp. 163–177.

[15] A. Griewank and A. Walther. *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. Second. Philadelphia, PA, USA: SIAM, 2008. ISBN: 0898714516.

[16] P. Guillaume and M. Masmoudi. "Computation of high order derivatives in optimal shape design". In: *Numerische Mathematik* 67.2 (1994), pp. 231–250.

[17] M. D. Gunzburger. *Perspectives in Flow Control and Optimization*. Advances in Design and Control. Philadelphia, PA, USA: SIAM, 2003. ISBN: 089871527X.

[18] M. Hinze and K. Kunisch. "Second Order Methods for Optimal Control of Time-Dependent Fluid Flow". In: *SIAM Journal on Control and Optimization* 40.3 (2001), pp. 925–946. DOI: `10.1137/S0363012999361810`.

[19] M. Hinze et al. *Optimization with PDE constraints*. Vol. 23. Mathematical Modelling: Theory and Applications. Berlin, Heidelberg, New York: Springer-Verlag, 2009. ISBN: 978-1-4020-8838-4.

[20] A. Jameson. "Aerodynamic design via control theory". In: *Journal of Scientific Computing* 3.3 (1988), pp. 233–260.

[21] Robert C Kirby. "From functional analysis to iterative methods". In: *SIAM review* 52.2 (2010), pp. 269–293.

[22] H.P. Langtangen and A. Logg. *Solving PDEs in Python: The FEniCS Tutorial I*. Simula SpringerBriefs on Computing. Springer International Publishing, 2017.

[23] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*. Vol. 84. Springer Science & Business Media, 2012.

[24] J. N. Lyness and C. B. Moler. “Numerical differentiation of analytic functions”. In: *SIAM Journal on Numerical Analysis* 4.2 (1967), pp. 202–210.

[25] D. Menemenlis et al. “ECCO2: High resolution global ocean and sea ice data synthesis”. In: *Mercator Ocean Quarterly Newsletter* 31 (2008), pp. 13–21. DOI: 10.1007/s10236-006-0082-1..

[26] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science+Business Media, LLC, 2006.

[27] J. Nocedal and S. J. Wright. *Numerical optimization*. 2nd ed. Berlin, Heidelberg, New-York: Springer-Verlag, 1999.

[28] M Nordaas and Simon W. Funke. *The Moola optimisation package*. https://github.com/funsim/moola. 2016.

[29] Rene Pinnau and Michael Ulbrich. *Optimization with PDE constraints*. Vol. 23. Springer Science & Business Media, 2008.

[30] R. L. Raffard and C. J. Tomlin. “Second order adjoint-based optimization of ordinary and partial differential equations with application to air traffic flow”. In: *Proceedings of the 2005 American Control Conference*. 2005, pp. 798–803. DOI: 10.1109/ACC.2005.1470057.

[31] Florian Rathgeber et al. “Firedrake: Automating the Finite Element Method by Composing Abstractions”. In: *ACM Trans. Math. Softw.* 43.3 (Dec. 2016), 24:1–24:27. ISSN: 0098-3500. DOI: 10.1145/2998441. URL: http://doi.acm.org/10.1145/2998441.

[32] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[33] Tobias Schwedes, David Ham, and Simon W Funke. *Code for simulations in Chapter 1 of ResearchBrief*. Dec. 2016. DOI: 10.5281/zenodo.223099. URL: https://doi.org/10.5281/zenodo.223099.

[34] Detlef Stammer and Carl Wunsch. *Computational requirements for ECCO in support of CLIVAR and GODAE*. The ECCO Report Series 2. Boston, USA: Scripps Institution of Oceanography, Massachusetts Institute of Technology, 1999.

[35] D. Stammer et al. “Ocean state estimation and prediction in support of oceanographic research”. In: *Oceanography* 2.13 (2000), pp. 51–56.

[36] D. Thevenin and G. Janiga. *Optimization and computational fluid dynamics*. Berlin, Heidelberg, New-York: Springer-Verlag, 2008. DOI: 10.1007/978-3-540-72153-6.

[37] Philip Wolfe. “Convergence Conditions for Ascent Methods”. In: *SIAM Review* 11.2 (1969), pp. 226–235.

[38] Margaret Wright. “The interior-point revolution in optimization: history, recent developments, and lasting consequences”. In: *Bulletin of the American mathematical society* 42.1 (2005), pp. 39–56.

http://www.springer.com/978-3-319-59482-8

Mesh Dependence in PDE-Constrained Optimisation
An Application in Tidal Turbine Array Layouts
Schwedes, T.; Ham, D.A.; Funke, S.W.; Piggott, M.D.
2017, VIII, 110 p. 24 illus., 21 illus. in color., Softcover
ISBN: 978-3-319-59482-8